IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
TYLER DIVISION

FILED
U.S. DISTRICT COURT
EASTERN DISTRICT OF TEXAS
JAN 12 2004
DAVID MALAND, CLERK
By
Deputy [illegible]

SOVERAIN SOFTWARE LLC,	)	
Plaintiff,	)	
v.	)	Civil Action No. 6-04CV-14
AMAZON.COM, INC., and THE GAP, INC.	)	JURY TRIAL DEMANDED
Defendants.	)	

COMPLAINT FOR PATENT INFRINGEMENT

Plaintiff Soverain Software LLC, for its Complaint against Defendants Amazon.com Inc., and The Gap, Inc. (collectively "Defendants"), alleges as follows.

INTRODUCTION

1. This is an action arising under 35 U.S.C. § 271 for Defendants' infringement of one or more of Soverain Software LLC's United States Patent Nos. 5,708,780, 5,715,314 and 5,909,492 (collectively, the "Patents in Suit").

THE PARTIES

2. Plaintiff Soverain Software LLC ("Soverain") is a Delaware limited liability company with its principal place of business at 120 South Riverside Plaza, Suite 430, Chicago, IL 60606.

3. Defendant Amazon.com, Inc. ("Amazon") is a Delaware corporation with its principal place of business at 1200 12th Avenue, Suite 1200, Seattle, Washington 98144. Amazon.com does business principally through the website www.amazon.com. Defendant Amazon.com, Inc. may be served by serving the Secretary of State of the State of Texas pursuant to the Texas Long Arm Statute, Texas Civil Practice & Remedies Code §§ 17.044 and asking the

Secretary of State to serve Amazon.com, Inc. at its principal place of business at 1200 12th Avenue, Suite 1200, Seattle, WA 98144 via Certified Mail - Return Receipt Requested.

4. Defendant The Gap, Inc. ("Gap") is a Delaware corporation with its principal place of business at Two Folsom Street, San Francisco, CA 94105. Defendant Gap does business principally through retail stores including those known as The Gap and through websites, including www.gap.com. Defendant The Gap, Inc. may be served by serving its registered agent for service, Corporation Service Company, 701 Brazos Street, Suite 1050, Austin, TX 78701.

JURISDICTION AND VENUE

5. This action arises under the patent laws of the United States, Title 35, United States Code. The jurisdiction of this Court over the subject matter of this action is proper under 28 U.S.C. § 1338.

6. Venue is proper in this Court pursuant to 28 U.S.C. §§ 1391(b) and (c) and 1400(b).

7. Personal jurisdiction exists over Defendants because they do business in Texas and infringing acts have occurred here.

PATENTS IN SUIT

8. Plaintiff Soverain is the owner of all right, title and interest in United States Patent No. 5,708,780 entitled "Internet Server Access Control And Monitoring Systems" ("the '780 patent"). The '780 patent was duly and properly issued by the United States Patent and Trademark Office on January 13, 1998. The '780 patent was assigned to Plaintiff Soverain. A copy the '780 patent is attached hereto as Exhibit A.

9. Plaintiff Soverain is the owner of all right, title and interest in United States Patent No. 5,715,314 entitled "Network Sales System" ("the '314 patent"). The '314 patent was

duly and properly issued by the United States Patent and Trademark Office on February 3, 1998. The '314 patent was assigned to Plaintiff Soverain. A copy the '314 patent is attached hereto as Exhibit B.

10. Plaintiff Soverain is the owner of all right, title and interest in United States Patent No. 5,909,492 entitled "Network Sales System" ("the '492 patent"). The '492 patent was duly and properly issued by the United States Patent and Trademark Office on June 1, 1999. The '492 patent was assigned to Plaintiff Soverain. A copy the '492 patent is attached hereto as Exhibit C.

11. Plaintiff Soverain has marked its products with the numbers of one or more of the Patents in Suit.

INFRINGEMENT OF THE PATENTS IN SUIT BY AMAZON

12. In violation of 35 U.S.C. § 271, Defendant Amazon has infringed and continues to infringe the '780, '314 and '492 patents by: (a) making, using, offering for sale or selling within the United States, or by importing into the United States, products or processes that practice inventions claimed in those patents; (b) inducing others to make, use, offer for sale or sell within the United States, or import into the United States, products or processes that practice inventions claimed in those patents; or (c) contributing to the making, using, offering for sale or selling within the United States, or importing into the United States, products or processes that practice inventions claimed in those patents.

13. On information and belief, Defendant Amazon has had notice of the '780, '314 and '492 patents at least as early as the filing of this Complaint.

14. On information and belief, the continued infringement by Defendant Amazon of the '780, '314 and '492 patents is deliberate and willful.

15. Plaintiff Soverain has been damaged by the infringement of its patents by Defendant Amazon and will continue to be damaged by such infringement.

16. Plaintiff Soverain has suffered and continues to suffer irreparable harm and will continue to do so unless Defendant Amazon is enjoined therefrom by this Court.

INFRINGEMENT OF THE PATENTS IN SUIT BY GAP

17. In violation of 35 U.S.C. § 271, Defendant Gap has infringed and continues to infringe the '314 and '492 patents by: (a) making, using, offering for sale or selling within the United States, or by importing into the United States, products or processes that practice inventions claimed in those patents; (b) inducing others to make, use, offer for sale or sell within the United States, or import into the United States, products or processes that practice inventions claimed in those patents; or (c) contributing to the making, using, offering for sale or selling within the United States, or importing into the United States, products or processes that practice inventions claimed in those patents.

18. On information and belief, Defendant Gap has had notice of the '314 and '492 patents at least as early as the filing of this Complaint.

19. On information and belief, the continued infringement by Defendant Gap of the '314 and '492 patents is deliberate and willful.

20. Plaintiff Soverain has been damaged by the infringement of its patents by Defendant Gap and will continue to be damaged by such infringement.

21. Plaintiff Soverain has suffered and continues to suffer irreparable harm and will continue to do so unless Defendant Gap is enjoined therefrom by this Court.

RELIEF REQUEST

Wherefore, Plaintiff Soverain respectfully requests that this Court enter judgment against Defendants Amazon and Gap as follows:

A. That each of the Patents in Suit is valid and enforceable;

B. That each of the '780, '314 and '492 patents has been infringed by Defendant Amazon.

C. That each of the '314 and '492 patents has been infringed by Defendant Gap.

D. That infringement of the Patents in Suit by Defendants Amazon and Gap has been willful;

E. An injunction against further infringement of the Patents in Suit by Defendants;

F. An award of damages adequate to compensate Plaintiff Soverain for the patent infringement that has occurred, together with pre-judgment interest and costs;

G. An award of all other damages permitted by 35 U.S.C. § 284, including increased damages up to three times the amount of compensatory damages found;

H. That this is an exceptional case and an award to Plaintiff Soverain of its costs and reasonable attorneys' fees incurred in this action as provided by 35 U.S.C. § 285; and

I. Such other relief as this Court deems just and proper.

THE ROTH LAW FIRM, P.C.

[illegible] MCS

Carl R. Roth
State Bar No. 17312000
115 N. Wellington, Suite 200
P.O. Box 876
Marshall, Texas 75761-1249
Telephone: 903-935-1665
Facsimile: 903-935-1797
Email: cr@rothfirm.com

Kenneth Adamo by perm. MCS

Kenneth R. Adamo
State Bar No. 00846960
JONES DAY
2727 North Harwood Street
Dallas, Texas 75201-1515
Telephone: 214-220-3939
Facsimile: 214-969-5100
Email: kradamo@jonesday.com

January 12, 2004

ATTORNEYS FOR SOVERAIN SOFTWARE LLC

Exhibit “A”

US005708780A

United States Patent [19]

Levergood et al.

[11] **Patent Number: 5,708,780**

[45] **Date of Patent: Jan. 13, 1998**

[54] **INTERNET SERVER ACCESS CONTROL AND MONITORING SYSTEMS**

[75] Inventors: **Thomas Mark Levergood**, Hopkinton; **Lawrence C. Stewart**, Burlington; **Stephen Jeffrey Morris**, Westford; **Andrew C. Payne**, Lincoln; **George Winfield Treese**, Newton, all of Mass.

[73] Assignee: **Open Market, Inc.**, Cambridge, Mass.

[21] Appl. No.: **474,096**

[22] Filed: **Jun. 7, 1995**

[51] **Int. Cl.**^{6} **G06F 15/56**

[52] **U.S. Cl.** **395/200.12**; 395/200.15

[58] **Field of Search** 395/200.11, 200.12, 395/200.02, 200.05, 200.06, 200.09, 200.15; 380/23, 24, 25, 49; 340/825.34

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,347,632	9/1994	Filepp et al.	395/200
5,544,322	8/1996	Cheng et al.	395/200.12
5,560,008	9/1996	Johnson et al.	395/200.09
5,577,209	11/1996	Boyle et al.	395/200.06

FOREIGN PATENT DOCUMENTS

0 456 920	11/1991	European Pat. Off. .	
0 645 688	3/1995	European Pat. Off. .	
WO 94/03859	2/1994	WIPO	G06F 13/14
WO 94/03959	2/1994	WIPO .	

OTHER PUBLICATIONS

Bina et al., Secure Access to Data over the Internet, 1994, pp. 99–102, IEEE.

Kiuchi et al., C–HTTP—the Development of a Secure, Closed HTTP based Network on the Internet, 1996, pp. 64–75, IEEE.

Ramanathan, Srinivas, et al., "Architectures for Personalized Multimedia," IEEE Multimedia, vol. 1, No. 1, Computer Society, pp. 37–46, 1994.

Choudhury, Abhijit K., et al., "Copyright Protection for Electronic Publishing Over Computer Networks," IEEE Network, The Magazine of Computer Communications, vol. 9, No. 3, pp. 12–20, May 1995.

Netscape Products, "Open and Secure Internet Software" Internet Sep. 18, 1995, pp. 1–2.

Merchant System: Overview, "Netscape Merchant System Data Sheet" Internet, Sep. 18, 1995, pp. 1–3.

Internet Applications Customer Showcase, "Customer Showcase" Internet, Sep. 18, 1995., pp. 1–2.

The Server–Application Function and Netscape Server API, "The Netscape Server API" Netscape Products Internet, Sep. 18, 1995, pp. 1–11.

The Object–Oriented Paradigm of Server Configuration, "The Object–Oriented Paradigm of Server Configuration" Internet, Sep. 18, 1995, p. 102.

Verisign Redirection Information, "Important Announcement" Internet, Sep. 18, 1995, p. 1.

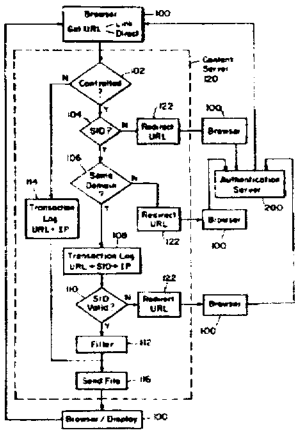
(List continued on next page.)

Primary Examiner—William M. Treat
Assistant Examiner—Patrice L. Winder
Attorney, Agent, or Firm—Hamilton, Brook, Smith & Reynolds, P.C.

[57] **ABSTRACT**

This invention relates to methods for controlling and monitoring access to network servers. In particular, the process described in the invention includes client-server sessions over the Internet involving hypertext files. In the hypertext environment, a client views a document transmitted by a content server with a standard program known as the browser. Each hypertext document or page contains links to other hypertext pages which the user may select to traverse. When the user selects a link that is directed to an access-controlled file, the server subjects the request to a secondary server which determines whether the client has an authorization or valid account. Upon such verification, the user is provided with a session identification which allows the user to access to the requested file as well as any other files within the present protection domain.

45 Claims, 7 Drawing Sheets



OTHER PUBLICATIONS

Lou Montulli, Electronic Mail to multiple recipients of the www–talk list (www–talk@www10.w3.org) on "Session Tracking" (omi.mail.www–talk, Apr. 18, 1995).

PR: Digital IDs for Open Market's Secure WebServer, "Press Release, VeriSign, Inc. to Provide Digital IDs for Open Market's Secure WebServer" Internet, Sep. 18, 1995, pp. 1–2.

PR: Online Security Solutions, "Verisign, Inc. Adds the Missing Component to Online Security Solutions" Internet, Sep. 18, 1995, pp. 1–2.

The SSL Protocol, Internet, Sep. 18, 1995, pp. 1–18.

IStore, "Netscape Istore Data Sheet" Internet, Sep. 18, 1995, pp. 1–2.

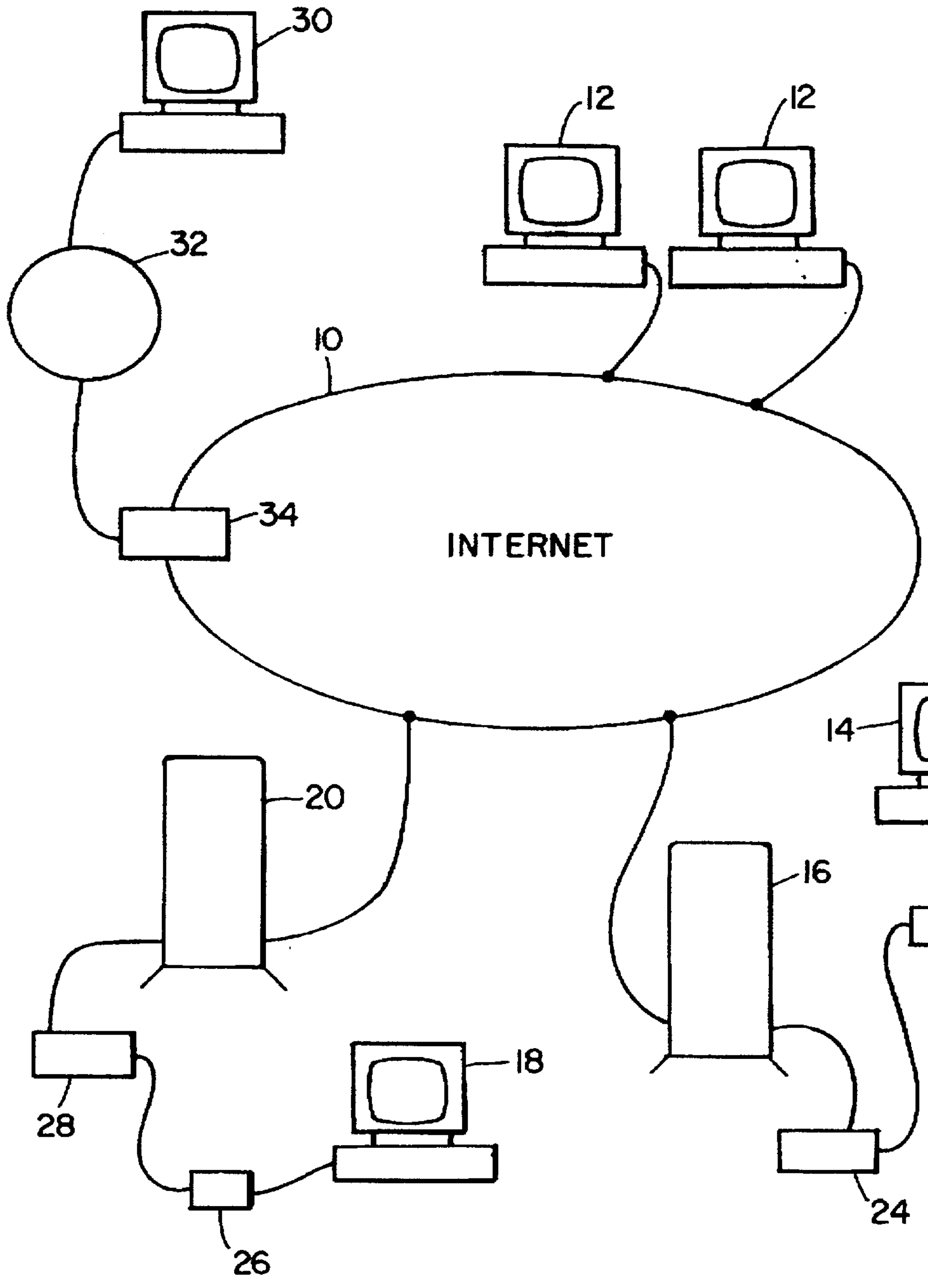


FIG. 1

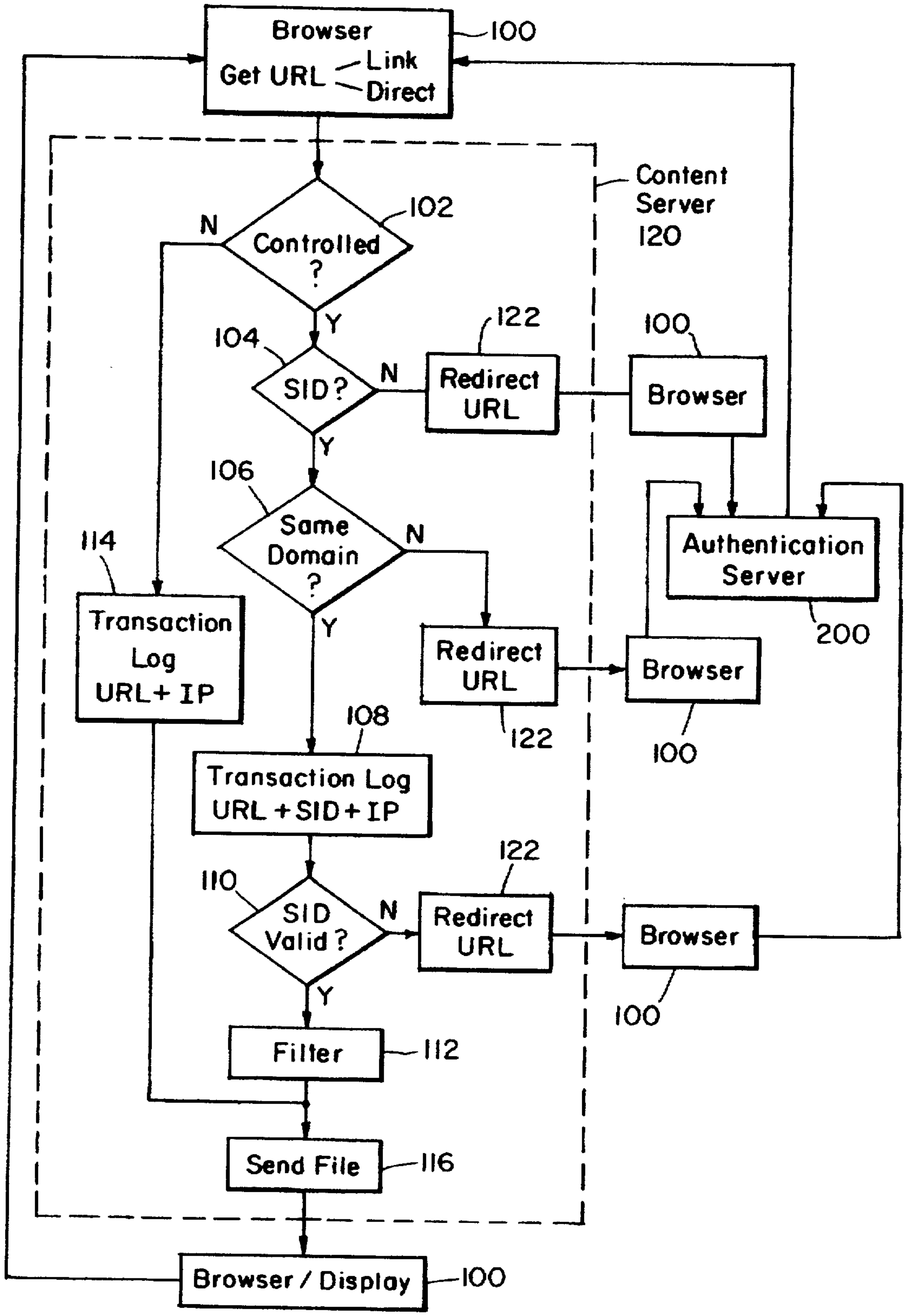

Browser
Get URL
Link
Direct
100
Content Server 120
Controlled ?
102
N
Y
104
SID?
N
Y
122
Redirect URL
100
Browser
106
Same Domain ?
N
Y
114
Transaction Log URL+ IP
Redirect URL
122
Browser
100
Authentication Server
200
108
Transaction Log URL + SID + IP
110
SID Valid ?
N
Y
122
Redirect URL
Browser
100
Filter
112
Send File
116
Browser / Display
100

FIG. 2A

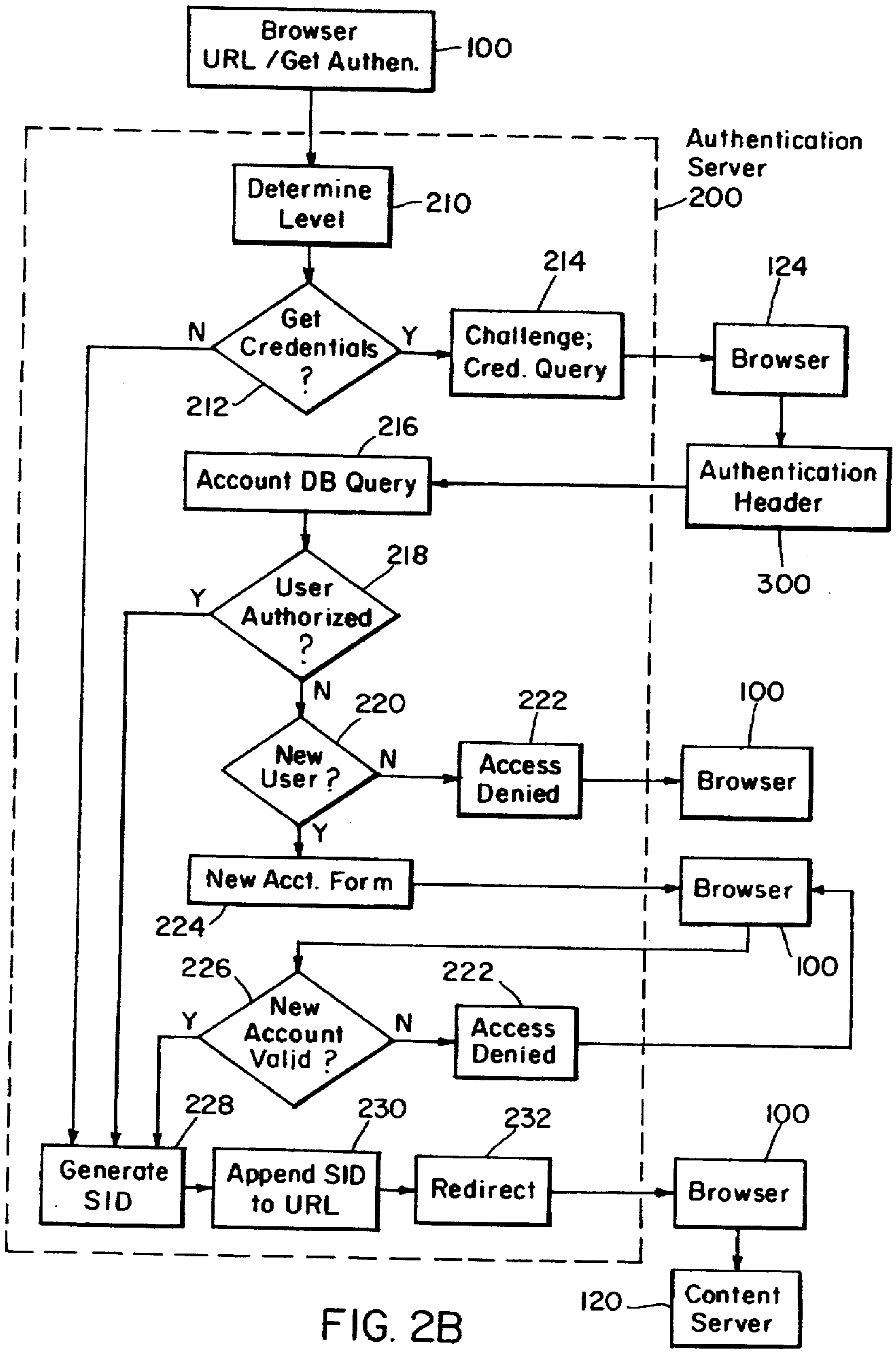

Browser URL /Get Authen.
100
Authentication Server
200
Determine Level
210
214
124
Get Credentials ?
N
Y
212
Challenge; Cred. Query
Browser
216
Account DB Query
Authentication Header
300
218
User Authorized ?
Y
N
220
222
100
New User ?
N
Y
Access Denied
Browser
New Acct. Form
224
Browser
100
226
222
New Account Valid ?
Y
N
Access Denied
228
230
232
100
Generate SID
Append SID to URL
Redirect
Browser
120
Content Server

FIG. 2B

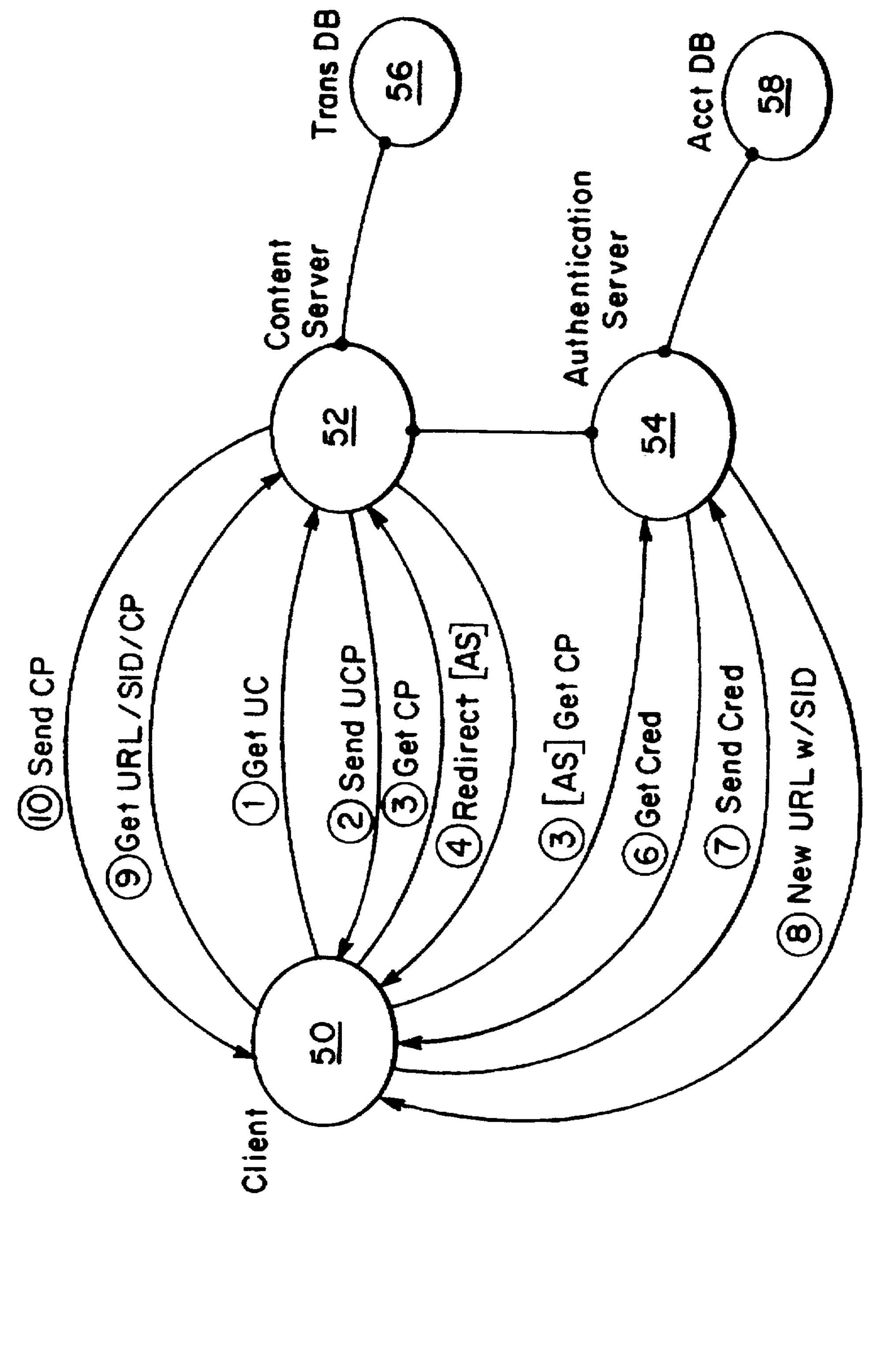

Client
50
Content Server
52
Trans DB
56
Authentication Server
54
Acct DB
58
① Get UC
② Send UCP
③ Get CP
④ Redirect [AS]
③ [AS] Get CP
⑥ Get Cred
⑦ Send Cred
⑧ New URL w/SID
⑨ Get URL/SID/CP
⑩ Send CP

FIG. 3

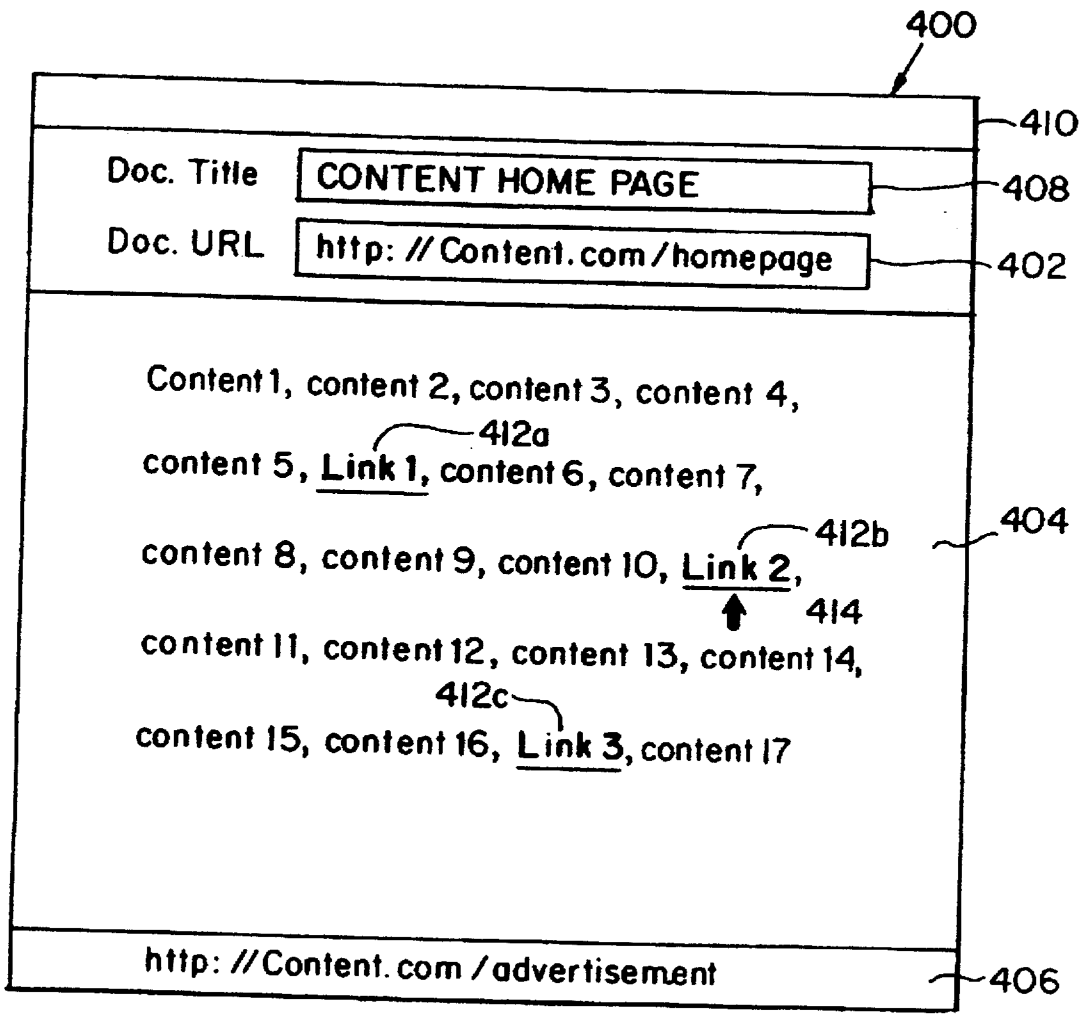
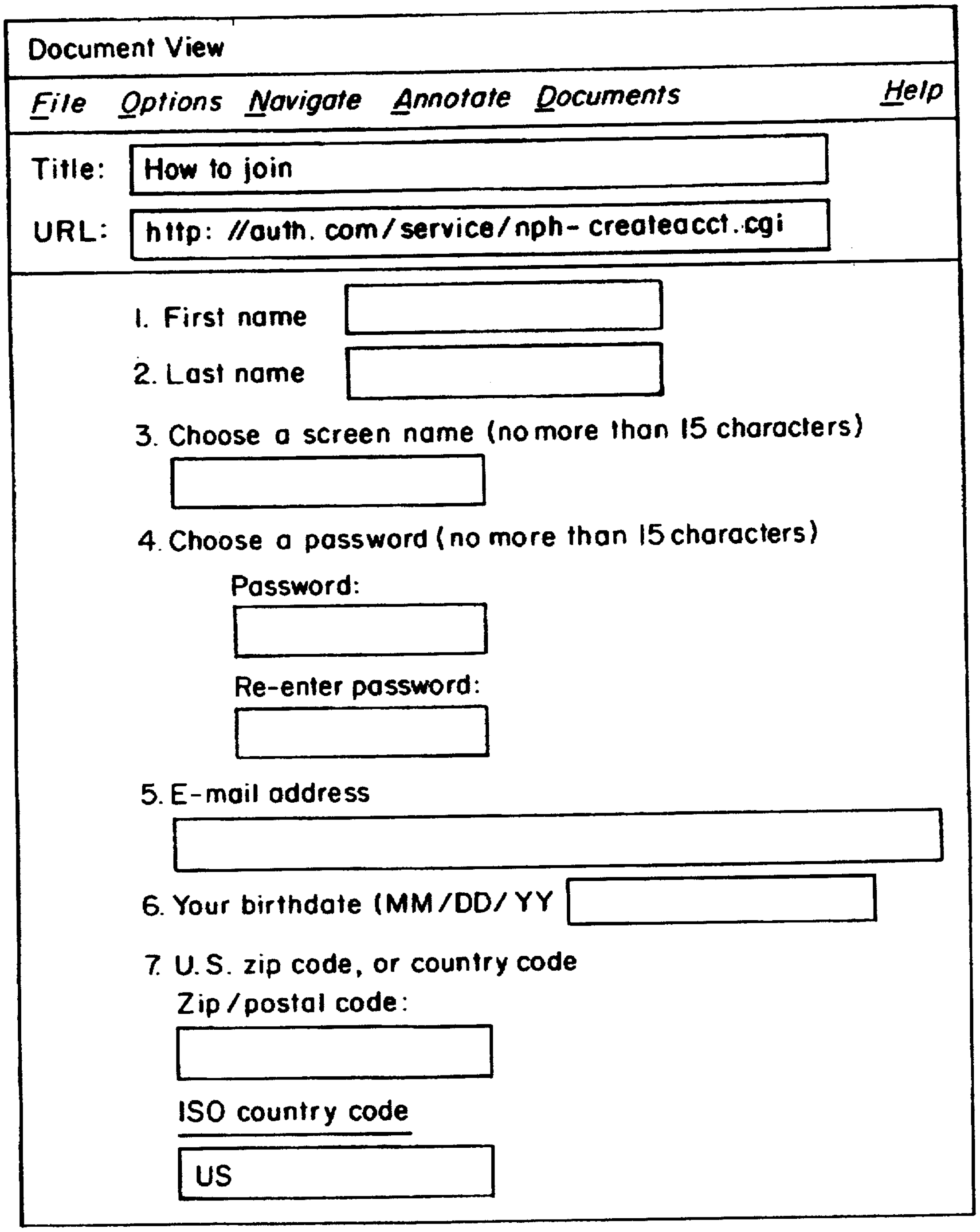

400
410
Doc. Title
CONTENT HOME PAGE
408
Doc. URL
http: //Content.com/homepage
402
Content 1, content 2, content 3, content 4,
412a
content 5, Link 1, content 6, content 7,
412b
404
content 8, content 9, content 10, Link 2,
414
content 11, content 12, content 13, content 14,
412c
content 15, content 16, Link 3, content 17
http: //Content.com /advertisement
406

FIG. 4



Title: How to join

URL: http: //auth. com/service/nph- createacct.cgi

1. First name

2. Last name

3. Choose a screen name (no more than 15 characters)

4. Choose a password (no more than 15 characters)

Password:

Re-enter password:

5. E-mail address

6. Your birthdate (MM/DD/ YY

7. U.S. zip code, or country code

Zip /postal code:

ISO country code

US

FIG. 5

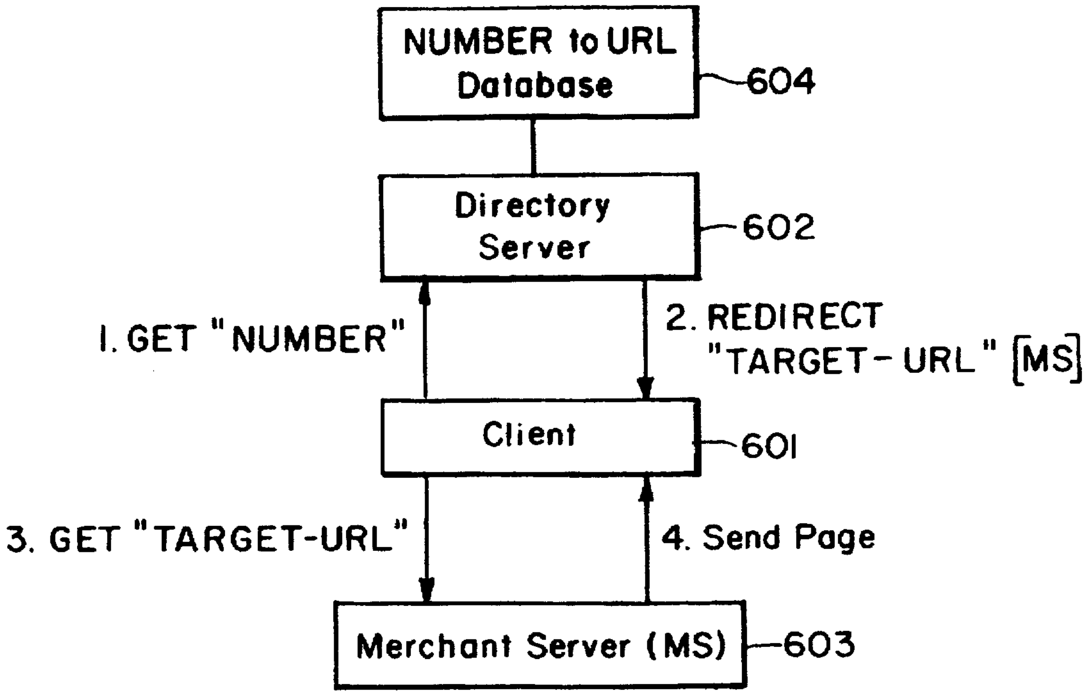

NUMBER to URL Database
604
Directory Server
602
1. GET "NUMBER"
2. REDIRECT "TARGET-URL" [MS]
Client
601
3. GET "TARGET-URL"
4. Send Page
Merchant Server (MS)
603

FIG. 6

INTERNET SERVER ACCESS CONTROL AND MONITORING SYSTEMS

REFERENCE TO APPENDIX

BACKGROUND OF THE INVENTION

The Internet, which started in the late 1960s, is a vast computer network consisting of many smaller networks that span the entire globe. The Internet has grown exponentially, and millions of users ranging from individuals to corporations now use permanent and dial-up connections to use the Internet on a daily basis worldwide. The computers or networks of computers connected within the Internet, known as "hosts", allow public access to databases featuring information in nearly every field of expertise and are supported by entities ranging from universities and government to many commercial organizations.

The information on the Internet is made available to the public through "servers". A server is a system running on an Internet host for making available files or documents contained within that host. Such files are typically stored on magnetic storage devices, such as tape drives or fixed disks, local to the host. An Internet server may distribute information to any computer that requests the files on a host. The computer making such a request is known as the "client", which may be an Internet-connected workstation, bulletin board system or home personal computer (PC).

TCP/IP (Transmission Control Protocol/Internet Protocol) is one networking protocol that permits full use of the Internet. All computers on a TCP/IP network need unique ID codes. Therefore, each computer or host on the Internet is identified by a unique number code, known as the IP (Internet Protocol) number or address, and corresponding network and computer names. In the past, an Internet user gained access to its resources only by identifying the host computer and a path through directories within the host's storage to locate a requested file. Although various navigating tools have helped users to search resources on the Internet without knowing specific host addresses, these tools still require a substantial technical knowledge of the Internet.

The World-Wide Web (Web) is a method of accessing information on the Internet which allows a user to navigate the Internet resources intuitively, without IP addresses or other technical knowledge. The Web dispenses with command-line utilities which typically require a user to transmit sets of commands to communicate with an Internet server. Instead, the Web is made up of hundreds of thousands of interconnected "pages", or documents, which can be displayed on a computer monitor. The Web pages are provided by hosts running special servers. Software which runs these Web servers is relatively simple and is available on a wide range of computer platforms including PC's. Equally available is a form of client software, known as a Web "browser", which is used to display Web pages as well as traditional non-Web files on the client system. Today, the Internet hosts which provide Web servers are increasing at a rate of more than 300 per month, en route to becoming the preferred method of Internet communication.

Created in 1991, the Web is based on the concept of "hypertext" and a transfer method known as "HTTP" (Hypertext Transfer Protocol). HTTP is designed to run primarily over TCP/IP and uses the standard Internet setup, where a server issues the data and a client displays or processes it. One format for information transfer is to create documents using Hypertext Markup Language (HTML). HTML pages are made up of standard text as well as formatting codes which indicate how the page should be displayed. The Web client, a browser, reads these codes in order to display the page. The hypertext conventions and related functions of the world wide web are described in the appendices of U.S. patent application Ser. No. 08/328,133, filed on Oct. 24, 1994, by Payne et al. which is incorporated herein by reference.

Each Web page may contain pictures and sounds in addition to text. Hidden behind certain text, pictures or sounds are connections, known as "hypertext links" ("links"), to other pages within the same server or even on other computers within the Internet. For example, links may be visually displayed as words or phrases that may be underlined or displayed in a second color. Each link is directed to a web page by using a special name called a URL (Uniform Resource Locator). URLs enable a Web browser to go directly to any file held on any Web server. A user may also specify a known URL by writing it directly into the command line on a Web page to jump to another Web page.

The URL naming system consists of three parts: the transfer format, the host name of the machine that holds the file, and the path to the file. An example of a URL may be:

http://www.college.univ.edu/Adir/Bdir/Cdir/page.html,

where "http" represents the transfer protocol; a colon and two forward slashes (://) are used to separate the transfer format from the host name; "www.college.univ.edu" is the host name in which "www" denotes that the file being requested is a Web page; "/Adir/Bdir/Cdir" is a set of directory names in a tree structure, or a path, on the host machine; and "page.html" is the file name with an indication that the file is written in HTML.

The Internet maintains an open structure in which exchanges of information are made cost-free without restriction. The free access format inherent to the Internet, however, presents difficulties for those information providers requiring control over their Internet servers. Consider for example, a research organization that may want to make certain technical information available on its Internet server to a large group of colleagues around the globe, but the information must be kept confidential. Without means for identifying each client, the organization would not be able to provide information on the network on a confidential or preferential basis. In another situation, a company may want to provide highly specific service tips over its Internet server only to customers having service contracts or accounts.

Access control by an Internet server is difficult for at least two reasons. First, when a client sends a request for a file on a remote Internet server, that message is routed or relayed by a web of computers connected through the Internet until it reaches its destination host. The client does not necessarily know how its message reaches the server. At the same time, the server makes responses without ever knowing exactly who the client is or what its IP address is. While the server may be programmed to trace its clients, the task of tracing is often difficult, if not impossible. Secondly, to prevent unwanted intrusion into private local area networks (LAN), system administrators implement various data-flow control mechanisms, such as the Internet "firewalls", within their networks. An Internet firewall allows a user to reach the Internet anonymously while preventing intruders of the outside world from accessing the user's LAN.

SUMMARY OF THE INVENTION

The present invention relates to methods of processing service requests from a client to a server through a network. In particular the present invention is applicable to processing client requests in an HTTP (Hypertext Transfer Protocol) environment, such as the World-Wide Web (Web). One aspect of the invention involves forwarding a service request from the client to the server and appending a session identification (SID) to the request and to subsequent service requests from the client to the server within a session of requests. In a preferred embodiment, the present method involves returning the SID from the server to the client upon an initial service request made by the client. A valid SID may include an authorization identifier to allow a user to access controlled files.

In a preferred embodiment, a client request is made with a Uniform Resource Locator (URL) from a Web browser. Where a client request is directed to a controlled file without an SID, the Internet server subjects the client to an authorization routine prior to issuing the SID, the SID being protected from forgery. A content server initiates the authorization routine by redirecting the client's request to an authentication server which may be at a different host. Upon receiving a redirected request, the authentication server returns a response to interrogate the client and then issues an SID to a qualified client. For a new client, the authentication server may open a new account and issue an SID thereafter. A valid SID typically comprises a user identifier, an accessible domain, a key identifier, an expiration time such as date, the IP address of the user computer, and an unforgettable digital signature such as a cryptographic hash of all of the other items in the SID encrypted with a secret key. The authentication server then forwards a new request consisting of the original URL appended by the SID to the client in a REDIRECT. The modified request formed by a new URL is automatically forwarded by the client browser to the content server.

When the content server receives a URL request accompanied by an SID, it logs the URL with the SID and the user IP address in a transaction log and proceeds to validate the SID. When the SID is so validated, the content server sends the requested document for display by the client's Web browser.

In the preferred embodiment, a valid SID allows the client to access all controlled files within a protection domain without requiring further authorization. A protection domain is defined by the service provider and is a collection of controlled files of common protection within one or more servers.

When a client accesses a controlled Web page with a valid SID, the user viewing the page may want to traverse a link to view another Web page. There are several possibilities. The user may traverse a link to another page in the same path. This is called a "relative link". A relative link may be made either within the same domain or to a different domain. The browser on the client computer executes a relative link by rewriting the current URL to replace the old controlled page name with a new one. The new URL retains all portions of the old, including the SID, except for the new page name. If the relative link points to a page in the same protection domain, the SID remains valid, and the request is honored.

However, if the relative link points to a controlled page in a different protection domain, the SID is no longer valid, and the client is automatically redirected to forward the rewritten URL to the authentication server to update the SID. The updated or new SID provides access to the new domain if the user is qualified.

The user may also elect to traverse a link to a document in a different path. This is called an "absolute link". In generating a new absolute link, the SID is overwritten by the browser. In the preferred embodiment, the content server, in each serving of a controlled Web page within the domain, filters the page to include the current SID in each absolute URL on the page. Hence, when the user elects to traverse an absolute link, the browser is facilitated with an authenticated URL which is directed with its SID to a page in a different path. In another embodiment, the content server may forego the filtering procedure as above-described and redirect an absolute URL to the authentication server for an update.

An absolute link may also be directed to a controlled file in a different domain. Again, such a request is redirected to the authentication server for processing of a new SID. An absolute link directed to an uncontrolled file is accorded an immediate access.

In another embodiment, a server access control may be maintained by programming the client browser to store an SID or a similar tag for use in each URL call to that particular server. This embodiment, however, requires a special browser which can handle such communications and is generally not suitable for the standard browser format common to the Web.

Another aspect of the invention is to monitor the frequency and duration of access to various pages both controlled and uncontrolled. A transaction log within a content server keeps a history of each client access to a page including the link sequence through which the page was accessed. Additionally, the content server may count the client requests exclusive of repeated requests from a common client. Such records provide important marketing feedback including user demand, access pattern, and relationships between customer demographics and accessed pages and access patterns.

The above and other features of the invention including various novel details of construction and combinations of parts will now be more particularly described with reference to the accompanying drawings and pointed out in the claims. It will be understood that the particular devices and methods embodying the invention are shown by way of illustration only and not as limitations of the invention. The principles and features of this invention may be employed in varied and numerous embodiments without departing from the scope of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a diagram illustrating the Internet operation.

FIG. 2A is a flowchart describing the preferred method of Internet server access control and monitoring.

FIG. 2B is a related flowchart describing the details of the authentication process.

FIG. 3 illustrates an example of a client-server exchange session involving the access control and monitoring method of the present invention.

FIG. 4 is an example of a World Wide Web page.

FIG. 5 is an example of an authorization form page.

FIG. 6 is a diagram describing the details of the translation of telephone numbers to URLs.

DETAILED DESCRIPTION OF THE INVENTION

Referring now to the drawings, FIG. 1 is a graphical illustration of the Internet. The Internet **10** is a network of millions of interconnected computers **12** including systems owned by Internet providers **16** and information systems (BBS) **20** such as Compuserve or America Online. Individual or corporate users may establish connections to the Internet in several ways. A user on a home PC **14** may purchase an account through the Internet provider **16**. Using a modem **22**, the PC user can dial up the Internet provider to connect to a high speed modem **24** which, in turn, provides a full service connection to the Internet. A user **18** may also make a somewhat limited connection to the Internet through a BBS **20** that provides an Internet gateway connection to its customers.

FIG. 2A is a flowchart detailing the preferred process of the present invention and FIG. 4 illustrates a sample Web page displayed at a client by a browser. The page includes text **404** which includes underlined link text **412**. The title bar **408** and URL bar **402** display the title and URL of the current web page, respectively. As shown in FIG. 4, the title of the page is "Content Home Page" and the corresponding URL is "http://content.com/homepage". When a cursor **414** is positioned over link text **412***b*, the page which would be retrieved by clicking a mouse is typically identified in a status bar **406** which shows the URL for that link. In this example the status bar **406** shows that the URL for the pointed link **412***b* is directed to a page called "advertisement", in a commercial content server called "content". By clicking on the link text, the user causes the browser to generate a URL GET request at **100** in FIG. 2A. The browser forwards the request to a content server **120**, which processes the request by first determining whether the requested page is a controlled document **102**. If the request is directed to an uncontrolled page, as in "advertisement" page in this example, the content server records the URL and the IP address, to the extent it is available, in the transaction log **114**. The content server then sends the requested page to the browser **116** for display on the user computer **117**.

If the request is directed to a controlled page, the content server determines whether the URL contains an SID **102**. For example, a URL may be directed to a controlled page name "report", such as "http://content.com/report", that requires an SID. If no SID is present, as in this example, the content server sends a "REDIRECT" response **122** to the browser **100** to redirect the user's initial request to an authentication server **200** to obtain a valid SID. The details of the authentication process are described in FIG. 2B and will be discussed later, but the result of the process is an SID provided from the authentication server to the client. In the above example, a modified URL appended with an SID may be: "http://content.com/[SID]/report". The preferred SID is a sixteen character ASCII string that encodes 96 bits of SID data, 6 bits per character. It contains a 32-bit digital signature, a 16-bit expiration date with a granularity of one hour, a 2-bit key identifier used for key management, an 8-bit domain comprising a set of information files to which the current SID authorizes access, and a 22-bit user identifier. The remaining bits are reserved for expansion. The digital signature is a cryptographic hash of the remaining items in the SID and the authorized IP address which are encrypted with a secret key which is shared by the authentication and content servers.

If the initial GET URL contains a SID, the content server determines whether the request is directed to a page within the current domain **106**. If the request having a SID is directed to a controlled page of a different domain, the SID is no longer valid and, again, the user is redirected to the authentication server **122**.

If the request is for a controlled page within the current domain, the content server proceeds to log the request URL, tagged with SID, and the user IP address in the transaction log **108**. The content server then validates the SID **110**. Such validation includes the following list of checks: (1) the SID's digital signature is compared against the digital signature computed from the remaining items in the SID and the user IP address using the secret key shared by the authentication and content servers; (2) the domain field of the SID is checked to verify that it is within the domain authorized; and (3) the EXP field of the SID is checked to verify that it is later than the current time.

If the validation passes, the content server searches the page to be forwarded for any absolute URL links contained therein **112**, that is, any links directed to controlled documents in different content servers. The content server augments each absolute URL with the current SID to facilitate authenticated accesses across multiple content servers. The requested page as processed is then transmitted to the client browser for display **117**. The user viewing the requested Web page may elect to traverse any link on that page to trigger the entire sequence again **100**.

FIG. 2B describes the details of the authentication process. The content server may redirect the client to an authentication server. The REDIRECT URL might be: "http://auth.com/authenticate?domain=[domain]&URL=http://content.com/report". That URL requests authentication and specifies the domain and the initial URL. In response to the REDIRECT, the client browser automatically sends a GET request with the provided URL.

Whenever the content server redirects the client to the authentication server **200**, the authentication server initiates the authorization process by validating that it is for an approved content server and determining the level of authentication required for the access requested **210**. Depending on this level, the server may challenge the user **212** for credentials. If the request is for a low level document, the authentication may issue an appropriate SID immediately **228** and forego the credential check procedures. If the document requires credentials, the authentication server sends a "CHALLENGE" response which causes the client browser to prompt the user for credentials **214**. A preferred credential query typically consists of a request for user name and password. If the user is unable to provide a password, the access is denied. The browser forms an authorization header **300** from the information provided, and resends a GET request to the authentication server using the last URL along with an authorization header. For example, a URL of such a GET request may be: "http://auth.com/authenticate?domain=[domain]&URL=http://content.com/report and the authorization header may be:"AUTHORIZE: [authorization]".

Upon receiving the GET request, the authentication server queries an account database **216** to determine whether the user is authorized **218** to access the requested document. A preferred account database may contain a user profile which includes information for identifying purposes, such as client IP address and password, as well as user demographic information, such as user age, home address, hobby, or occupation, for later use by the content server. If the user is authorized, an SID is generated **228** as previously described. If the user is not cleared for authorization, the authentication server checks to see if the user qualifies for a new account **220**. If the user is not qualified to open a new account, a page denying access **222** is transmitted to the client browser **100**. If the user is qualified, the new user is sent a form page such as illustrated in FIG. **5** to initiate a real-time on-line registration **224**. The form may, for example, require personal information and credit references from the user. The browser is able to transmit the data entered by the user in the blanks **502** as a "POST" message to the authentication server. A POST message causes form contents to be sent to the server in a data body other than as part of the URL. If the registration form filled out by the new user is valid **226**, an appropriate SID is generated **228**. If the registration is not valid, access is again denied **222**.

An SID for an authorized user is appended ("tagged") **230** to the original URL directed to a controlled page on the content server. The authentication server then transmits a REDIRECT response **232** based on the tagged URL to the client browser **100**. The modified URL, such as "http://content.com/[SID]/report" is automatically forwarded to the content server **120**.

FIG. 3, illustrates a typical client-server exchange involving the access control and monitoring method of the present invention. In Step **1**, the client **50** running a browser transmits a GET request through a network for an uncontrolled page (UCP). For example, the user may request an advertisement page by transmitting a URL "http://content.com/advertisement", where "content.com" is the server name and "advertisement" is the uncontrolled page name. In Step **2**, the content server **52** processes the GET request and transmits the requested page, "advertisement". The content server also logs the GET request in the transaction database **56** by recording the URL, the client IP address, and the current time.

In Step **3**, the user on the client machine may elect to traverse a link in the advertisement page directed to a controlled page (CP). For example, the advertisement page may contain a link to a controlled page called "report". Selecting this link causes the client browser **50** to forward a GET request through a URL which is associated with the report file "http://content.com/report". The content server **52** determines that the request is to a controlled page and that the URL does not contain an SID. In Step **4**, the content server transmits a REDIRECT response to the client, and, in Step **5**, the browser automatically sends the REDIRECT URL to the authentication server **54**. The REDIRECT URL sent to the authentication server may contain the following string:

"http://auth.com/authenticate?domain=[domain]&URL=http://content.com/report".

The authentication server processes the REDIRECT and determines whether user credentials (CRED) are needed for authorization. In Step **6**, the authentication server transmits a "CHALLENGE" response to the client. As previously described, typical credentials consist of user name and password. An authorization header based on the credential information is then forwarded by the client browser to the authentication server. For example, a GET URL having such an authorization header is:

"http://autho.com/authenticate?domain=[domain]&URL=http://content.com/report and the authorization header may be: "AUTHORIZE:[authorization]". The authentication server processes the GET request by checking the Account Database **58**. If a valid account exists for the user, an SID is issued which authorizes access to the controlled page "report" and all the other pages within the domain.

As previously described, the preferred SID comprises a compact ASCII string that encodes a user identifier, the current domain, a key identifier, an expiration time, the client IP address, and an unforgeable digital signature. In Step **8**, the authentication server redirects the client to the tagged URL, "http://content.com/[SID]/report", to the client. In Step **9**, the tagged URL is automatically forwarded by the browser as a GET request to the content server. The content server logs the GET request in the Transaction database **56** by recording the tagged URL, the client IP address, and the current time. In Step **10**, the content server, upon validating the SID, transmits the requested controlled page "report" for display on the client browser.

According to one aspect of the present invention, the content server periodically evaluates the record contained in the transaction log **56** to determine the frequency and duration of accesses to the associated content server. The server counts requests to particular pages exclusive of repeated requests from a common client in order to determine the merits of the information on different pages for ratings purposes. By excluding repeated calls, the system avoids distortions by users attempting to "stuff the ballot box." In one embodiment, the time intervals between repeated requests by a common client are measured to exclude those requests falling within a defined period of time.

Additionally, the server may, at any given time, track access history within a client-server session. Such a history profile informs the service provider about link transversal frequencies and link paths followed by users. This profile is produced by filtering transaction logs from one or more servers to select only transactions involving a particular user ID (UID). Two subsequent entries, A and B, corresponding to requests from a given user in these logs represent a link traversal from document A to document B made by the user in question. This information may be used to identify the most popular links to a specific page and to suggest where to insert new links to provide more direct access. In another embodiment, the access history is evaluated to determine traversed links leading to a purchase of a product made within commercial pages. This information may be used, for example, to charge for advertising based on the number of link traversals from an advertising page to a product page or based on the count of purchases resulting from a path including the advertisement. In this embodiment, the server can gauge the effectiveness of advertising by measuring the number of sales that resulted from a particular page, link, or path of links. The system can be configured to charge the merchant for an advertising page based on the number of sales that resulted from that page.

According to another aspect of the present invention, a secondary server, such as the authentication server **200** in FIG. 2B, may access a prearranged user profile from the account database **216** and include information based on such a profile in the user identifier field of the SID. In a preferred embodiment, the content server may use such an SID to customize user requested pages to include personalized content based on the user identifier field of the SID.

In another aspect of the invention, the user may gain access to domain of servers containing journals or publications through a subscription. In such a situation, the user may purchase the subscription in advance to gain access to on-line documents through the Internet. The user gains access to a subscribed document over the Internet through the authorization procedure as described above where an authorization indicator is preferably embedded in a session identifier. In another embodiment, rather than relying on a prepaid subscription, a user may be charged and billed each time he or she accesses a particular document through the Internet. In that case, authorization may not be required so long as the user is fully identified in order to be charged for the service. The user identification is most appropriately embedded in the session identifier described above.

In another aspect of the invention, facilities are provided to allow users to utilize conventional telephone numbers or other identifiers to access merchant services. These merchant services can optionally be protected using SIDs. In a preferred embodiment, as shown in FIG. 6, a Web browser client **601** provides a "dial" command to accept a telephone number from a user, as by clicking on a "dial" icon and inputting the telephone number through the keyboard. The browser then constructs a URL of the form "http://directory.net/NUMBER", where NUMBER is the telephone number or other identifier specified by the user. The browser then performs a GET of the document specified by this URL, and contacts directory server **602**, sending the NUMBER requested in Message **1**.

In another embodiment, implemented with a conventional browser, client **601** uses a form page provided by directory server **602** that prompts for a telephone number or other identifier in place of a "dial" command, and Message **1** is a POST message to a URL specified by this form page.

Once NUMBER is received by directory server **602**, the directory server uses database **604** to translate the NUMBER to a target URL that describes the merchant server and document that implements the service corresponding to NUMBER. This translation can ignore the punctuation of the number, therefore embedded parenthesis or dashes are not significant.

In another embodiment an identifier other than a number may be provided. For example, a user may enter a company name or product name without exact spelling. In such a case a "soundex" or other phonetic mapping can be used to permit words that sound alike to map to the same target URL. Multiple identifiers can also be used, such as a telephone number in conjunction with a product name or extension.

In Message **2**, Directory server **602** sends a REDIRECT to client **601**, specifying the target URL for NUMBER as

computed from database **604**. The client browser **601** then automatically sends Message **3** to GET the contents of this URL. Merchant server **603** returns this information in Message **4**. The server **602** might have returned a Web page to the client to provide an appropriate link to the required document. However, because server **602** makes a translation to a final URL and sends a REDIRECT rather than a page to client **601**, the document of message **4** is obtained without any user action beyond the initial dial input.

The Target URL contained in Message **3** can be an ordinary URL to an uncontrolled page, or it can be a URL that describes a controlled page. If the Target URL describes a controlled page then authentication is performed as previously described. The Target URL can also describe a URL that includes an SID that provides a preauthorized means of accessing a controlled page.

Among benefits of the "dial" command and its implementation is an improved way of accessing the Internet that is compatible with conventional telephone numbers and other identifiers. Merchants do not need to alter their print or television advertising to provide an Internet specific form of contact information, and users do not need to learn about URLs.

In the approach a single merchant server can provide multiple services that correspond to different external "telephone numbers" or other identifiers. For example, if users dial the "flight arrival" number they could be directed to the URL for the arrival page, while, if they dial the "reservations" number, they would be directed to the URL for the reservations page. A "priority gold" number could be directed to a controlled page URL that would first authenticate the user as belonging to the gold users group, and then would provide access to the "priority gold" page. An unpublished "ambassador" number could be directed to a tagged URL that permits access to the "priority gold" page without user authentication.

Equivalents

Those skilled in the art will know, or be able to ascertain using no more than routine experimentation, many equivalents to the specific embodiments or the invention described herein. These and all other equivalents are intended to be encompassed by the following claims.

-30-

```
/
 * tclSidSup.c --
 *
 *      tcl SID pack/unpack/id routines
 *
 *  Steve Morris
 *  morris@openmarket.com
 */

/* #define debug 1 */

#include <stdio.h>
#include <sys/types.h>
#include <stdlib.h>
#include <unistd.h>
#ifdef __alpha
#include <sys/time.h>
#endif
#include <tcl.h>
#include "global.h"
#include "md5.h"



#ifdef debug
#define dbg_out(s) printf("%s\n", s);
#define dbg_outi(s,a) printf("%s %x\n", s,a);
#define dbg_outs(s,a) printf("%s %s\n", s,a);
#else
#define dbg_out(s)
#define dbg_outi(s,a)
#define dbg_outs(s,a)
#endif


/* external routines called
*/
      char * radix64decode_noslash(char *in, int len, int *output_len);
      char * radix64encode_noslash(char *in, int len);
      int compute_ihash(char *str);


#define         sid_rev_zero (0)


#define get_sid(lw,pos,mask) ((bsid[lw]>>pos) & mask)
#define put_sid(lw,pos,mask,data) {bsid[lw] &= ~(0xffffffff & (mask<<pos)); \
                                   bsid[lw] |= ((data & mask)<<pos); }


#define sig_lw      0
#define sig_pos     0
#define sig_mask    0xffffffff

#define kid_lw      1
#define kid_pos     0
#define kid_mask    0x3
#define rev_lw      1
#define rev_pos     2
#define rev_mask    0x3
#define uid_lw      1
```

2

```
#..efine uid_pos      4
#define uid_mask      0x3fffff
#define rsrv1_lw      1
#define rsrv1_pos     26
#define rsrv1_mask    0x3f


#define dom_lw        2
#define dom_pos       0
#define dom_mask      0xff
#define rsrv2_lw      2
#define rsrv2_pos     8
#define rsrv2_mask    0x3f
#define uctx_lw       2
#define uctx_pos      14
#define uctx_mask     0x3
#define exp_lw        2
#define exp_pos       16
#define exp_mask      0xffff


#ifdef __sun__
#define fix_endian(i,o)    {  ecp = (char *) i; \
                              eda  = (*ecp++)<<0; \
                              eda |= (*ecp++)<<8; \
                              eda |= (*ecp++)<<16; \
                              eda |= (*ecp)<<24; \
                              (int *) *o = eda;}
#else
/* #define fix_endian(i,o)    { (int *) *o = (int *) *i;} */
#define fix_endian(i,o)
#endif
```

```
/* TclUnpackSid[NoValidate]
 *    Converts a binary sid to the internal representation, and does minimal
 *    validation on the SID, the NoValidate version does even less.
 * Inputs:
 *    the sid          a sid, looks like "/@@u9GIg4QEAAAAABPn"
 * Returns:
 *    {%d %d %d %d}    on succes (dom uid keyid exp)
 * or ""               on failure
 *
 */


int TclUnpackSid(ClientData dummy, Tcl_Interp *interp,
                        int argc, char **argv)
    {

    char temp[512];
    int ret_status = 1;
    int i;
    int * bsid = NULL, act_hash;
    char ip_address[32]="", secret[200]="", *cp;
    unsigned char *ecp;
    unsigned int eda;
    char hash_buffer[512];

    struct sid_info
      {
      unsigned int sig;
      unsigned int kid;
      unsigned int uid;
      unsigned int rev;
      unsigned int dom;
      unsigned int exp;
      unsigned int uctx;
      } esid;

     if (argc != 2)
       {
       Tcl_AppendResult(interp, "wrong # args: should be \"", argv[0],
         " string\"", (char *) NULL);
       return TCL_ERROR;
       }

 #ifdef test_ip
     strcpy(ip_address, test_ip);
 #endif
 #ifdef test_secret
     strcpy(secret, test_secret);
 #endif

       cp = Tcl_GetVar(interp, "sidip", TCL_GLOBAL_ONLY|TCL_LEAVE_ERR_MSG);
       if (cp != NULL) strncpy(ip_address, cp, 31);

       cp = argv[1]+3;
       i = strlen(cp);
       if (i != 16) goto sid_bad;
```

-33-

```
/* first convert the SID back to binary*/
bsid = (int *) radix64decode_noslash(cp, i, &i);
if (bsid == NULL) goto sid_bad;
if (i != 12) goto sid_bad;

fix_endian(&bsid[0], &bsid[0]);
fix_endian(&bsid[1], &bsid[1]);
fix_endian(&bsid[2], &bsid[2]);

dbg_outi("sid[0]=",bsid[0]);
dbg_outi("sid[1]=",bsid[1]);
dbg_outi("sid[2]=",bsid[2]);

dbg_outi("sig =",get_sid(sig_lw,sig_pos,sig_mask));
dbg_outi("kid =",get_sid(kid_lw,kid_pos,kid_mask));
dbg_outi("uid =",get_sid(uid_lw,uid_pos,uid_mask));
dbg_outi("rev =",get_sid(rev_lw,rev_pos,rev_mask));
dbg_outi("dom =",get_sid(dom_lw,dom_pos,dom_mask));
dbg_outi("exp =",get_sid(exp_lw,exp_pos,exp_mask));
dbg_outi("uctx =",get_sid(uctx_lw,uctx_pos,uctx_mask));
dbg_outi("rs1 =",get_sid(rsrv1_lw,rsrv1_pos,rsrv1_mask));
dbg_outi("rs2 =",get_sid(rsrv2_lw,rsrv2_pos,rsrv2_mask));


/* check the SID version field */
if (get_sid(rev_lw,rev_pos,rev_mask) != sid_rev_zero) goto sid_bad;
if (get_sid(rsrv1_lw,rsrv1_pos,rsrv1_mask) != 0) goto sid_bad;
if (get_sid(rsrv2_lw,rsrv2_pos,rsrv2_mask) != 0) goto sid_bad;

/* unpack the sid */
esid.sig = get_sid(sig_lw,sig_pos,sig_mask);
esid.kid = get_sid(kid_lw,kid_pos,kid_mask);
esid.uid = get_sid(uid_lw,uid_pos,uid_mask);
esid.rev = get_sid(rev_lw,rev_pos,rev_mask);
esid.dom = get_sid(dom_lw,dom_pos,dom_mask);
esid.exp = get_sid(exp_lw,exp_pos,exp_mask);
esid.uctx = get_sid(uctx_lw,uctx_pos,uctx_mask);

dbg_outi("sig =",esid.sig);
dbg_outi("kid =",esid.kid);
dbg_outi("uid =",esid.uid);
dbg_outi("rev =",esid.rev);
dbg_outi("dom =",esid.dom);
dbg_outi("exp =",esid.exp);
dbg_outi("uctx =",esid.uctx);

sprintf(temp,"secret(%d)",esid.kid);
cp = Tcl_GetVar(interp, temp, TCL_GLOBAL_ONLY);
if (cp != NULL) strncpy(secret, cp, 199);

/* hash the sid and check the signature */
sprintf(hash_buffer, "%s%s%08x%08x", secret,ip_address,bsid[2],bsid[1]);
dbg_outs("hashing ->",hash_buffer);
act_hash = compute_ihash(hash_buffer);
fix_endian(&act_hash, &act_hash);
dbg_outi("expected hash=",bsid[0]);
dbg_outi("actual   hash=",act_hash);
if (act_hash != esid.sig) goto  sid_bad;

rtn_exit:
      if (bsid != NULL) free(bsid);
      if (ret_status) sprintf(interp->result,"{%d %d %d %d %d}",
          esid.dom,esid.uid,esid.kid,esid.exp,esid.uctx);
      else sprintf(interp->result,"");
      return TCL_OK;


    sid_bad:
      ret_status = 0;
      goto rtn_exit;

   exp_bad:
      ret_status = 0;
      goto rtn_exit;


}
```

```
int TclUnpackSidNoValidate(ClientData dummy, Tcl_Interp *interp,
                           int argc, char **argv)
    {

    char temp[512];
    int ret_status = 1;
    int i;
    int * bsid = NULL, act_hash;
    char ip_address[32]="", secret[200]="", *cp;
    unsigned char *ecp;
    unsigned int eda;
    char hash_buffer[512];

    struct sid_info
      {
      unsigned int sig;
      unsigned int kid;
      unsigned int uid;
      unsigned int rev;
      unsigned int dom;
      unsigned int exp;
      unsigned int uctx;
      } esid;

    if (argc != 2)
      {
      Tcl_AppendResult(interp, "wrong # args: should be \"", argv[0],
        " string\"", (char *) NULL);
      return TCL_ERROR;
      }

#ifdef test_ip
    strcpy(ip_address, test_ip);
#endif
#ifdef test_secret
    strcpy(secret, test_secret);
#endif

      cp = Tcl_GetVar(interp, "sidip", TCL_GLOBAL_ONLY|TCL_LEAVE_ERR_MSG);
      if (cp != NULL) strncpy(ip_address, cp, 31);

      cp = argv[1]+3;
      i = strlen(cp);
      if (i != 16) goto sid_bad;

      /* first convert the SID back to binary*/
      bsid = (int *) radix64decode_noslash(cp, i, &i);
      if (bsid == NULL) goto sid_bad;
      if (i != 12) goto sid_bad;

      fix_endian(&bsid[0], &bsid[0]);
      fix_endian(&bsid[1], &bsid[1]);
      fix_endian(&bsid[2], &bsid[2]);

      dbg_outi("sid[0]=",bsid[0]);
      dbg_outi("sid[1]=",bsid[1]);
      dbg_outi("sid[2]=",bsid[2]);

      dbg_outi("sig =",get_sid(sig_lw,sig_pos,sig_mask));
```

```
        dbg_outi("kid =",get_sid(kid_lw,kid_pos,kid_mask));
        dbg_outi("uid =",get_sid(uid_lw,uid_pos,uid_mask));
        dbg_outi("rev =",get_sid(rev_lw,rev_pos,rev_mask));
        dbg_outi("dom =",get_sid(dom_lw,dom_pos,dom_mask));
        dbg_outi("exp =",get_sid(exp_lw,exp_pos,exp_mask));
        dbg_outi("uctx =",get_sid(uctx_lw,uctx_pos,uctx_mask));
        dbg_outi("rs1 =",get_sid(rsrv1_lw,rsrv1_pos,rsrv1_mask));
        dbg_outi("rs2 =",get_sid(rsrv2_lw,rsrv2_pos,rsrv2_mask));


        /* check the SID version field */
        if (get_sid(rev_lw,rev_pos,rev_mask) != sid_rev_zero) goto sid_bad;
        if (get_sid(rsrv1_lw,rsrv1_pos,rsrv1_mask) != 0) goto sid_bad;
        if (get_sid(rsrv2_lw,rsrv2_pos,rsrv2_mask) != 0) goto sid_bad;

        /* unpack the sid */
        esid.sig = get_sid(sig_lw,sig_pos,sig_mask);
        esid.kid = get_sid(kid_lw,kid_pos,kid_mask);
        esid.uid = get_sid(uid_lw,uid_pos,uid_mask);
        esid.rev = get_sid(rev_lw,rev_pos,rev_mask);
        esid.dom = get_sid(dom_lw,dom_pos,dom_mask);
        esid.exp = get_sid(exp_lw,exp_pos,exp_mask);
        esid.uctx = get_sid(uctx_lw,uctx_pos,uctx_mask);

        dbg_outi("sig =",esid.sig);
        dbg_outi("kid =",esid.kid);
        dbg_outi("uid =",esid.uid);
        dbg_outi("rev =",esid.rev);
        dbg_outi("dom =",esid.dom);
        dbg_outi("exp =",esid.exp);
        dbg_outi("uctx =",esid.uctx);

#if (1 == 0) /* disable validation for the novalidate case */
        sprintf(temp,"secret(%d)",esid.kid);
        cp = Tcl_GetVar(interp, temp, TCL_GLOBAL_ONLY);
        if (cp != NULL) strncpy(secret, cp, 199);

       /* hash the sid and check the signature */
       sprintf(hash_buffer, "%s%s%08x%08x", secret,ip_address,bsid[2],bsid[1]);
       dbg_outs("hashing ->",hash_buffer);
       act_hash = compute_ihash(hash_buffer);
       fix_endian(&act_hash, &act_hash);
       dbg_outi("expected hash=",bsid[0]);
       dbg_outi("actual   hash=",act_hash);
       if (act_hash != esid.sig) goto  sid_bad;
#endif

      rtn_exit:
        if (bsid != NULL) free(bsid);
        if (ret_status) sprintf(interp->result,"{%d %d %d %d %d}",
            esid.dom,esid.uid,esid.kid,esid.exp, esid.uctx);
        else sprintf(interp->result,"");
        return TCL_OK;


      sid_bad:
        ret_status = 0;
        goto rtn_exit;

     exp_bad:
        ret_status = 0;
        goto rtn_exit;

}
```

```
/* TclPackSid
 *    Creates the ascii representation of a binary sid
 * Inputs:
 *    "{%d %d %d %d [%d]}  Dom uid keyid exp [uctx]
 * Returns:
 *    ascii bin_sid like  "/@@u9GIg4QEAAAAABPn"
 *
 */


int TclPackSid(ClientData dummy, Tcl_Interp *interp,
                         int argc, char **argv)
    {
    char temp[512];
    int bsid[3] = {0,0,0}, i, act_hash;
    unsigned char *ecp;
    unsigned int eda;

    char ip_address[32], secret[200]="", *cp;
    int dom, uid, keyid, exp, uctx;
    char hash_buffer[512];

    struct sid_info
      {
      unsigned int sig;
      unsigned int kid;
      unsigned int uid;
      unsigned int rev;
      unsigned int dom;
      unsigned int exp;
      unsigned int uctx;
      } esid;

    if (argc != 2)
      {
      Tcl_AppendResult(interp, "wrong # args: should be \"", argv[0],
        " string\"", (char *) NULL);
      return TCL_ERROR;
      }


     i = sscanf(argv[1],"%d %d %d %d %d", &dom, &uid, &keyid, &exp, &uctx);
     dbg_outs("scanning...",argv[1]);
     dbg_outi("scan returned ",i);
     if (i == 4) {uctx = 0; i = 5;}
     if (i != 5) return TCL_ERROR;

      esid.kid = keyid;
      esid.uid = uid;
      esid.dom = dom;
      esid.exp = exp;
      esid.uctx = uctx;
      dbg_outi("kid=",esid.kid);
      dbg_outi("uid=",esid.uid);
      dbg_outi("dom=",esid.dom);
      dbg_outi("exp=",esid.exp);
      dbg_outi("uctx=",esid.uctx);
#ifdef test_ip
    strcpy(ip_address, test_ip);
#endif
#ifdef test_secret
    strcpy(secret, test_secret);
#endif
        cp = Tcl_GetVar(interp, "sidip", TCL_GLOBAL_ONLY);
        if (cp != NULL) strncpy(ip_address, cp, 31);


        sprintf(temp,"secret(%d)",esid.kid);
        cp = Tcl_GetVar(interp, temp, TCL_GLOBAL_ONLY);
        if (cp != NULL) strncpy(secret, cp, 199);


        put_sid(kid_lw,   kid_pos,   kid_mask,   esid.kid);
        put_sid(uid_lw,   uid_pos,   uid_mask,   esid.uid);
        put_sid(rsrv1_lw, rsrv1_pos, rsrv1_mask, 0);
        put_sid(rev_lw,   rev_pos,   rev_mask,   sid_rev_zero);
        put_sid(dom_lw,   dom_pos,   dom_mask,   esid.dom);
        put_sid(rsrv2_lw, rsrv2_pos, rsrv2_mask, 0);
        put_sid(exp_lw,   exp_pos,   exp_mask,   esid.exp);
        put_sid(uctx_lw,  uctx_pos,  uctx_mask,  esid.uctx);

        dbg_outi("sid[0]=",bsid[0]);
        dbg_outi("sid[1]=",bsid[1]);
        dbg_outi("sid[2]=",bsid[2]);

        sprintf(hash_buffer, "%s%s%08x%08x", secret,ip_address,bsid[2],bsid[1]);
        dbg_outs("ascii sid built ",hash_buffer);
        act_hash = compute_ihash(hash_buffer);
/*      fix_endian(&act_hash, &act_hash); */
        put_sid(sig_lw, sig_pos, sig_mask, act_hash);

/*      dbg_outi("sid[0]=",bsid[0]);
        dbg_outi("sid[1]=",bsid[1]);
        dbg_outi("sid[2]=",bsid[2]);    */


/*      fix_endian(&bsid[0], &bsid[0]); */
        fix_endian(&bsid[1], &bsid[1]);
        fix_endian(&bsid[2], &bsid[2]);

        dbg_outi("sid[0]=",bsid[0]);
        dbg_outi("sid[1]=",bsid[1]);
        dbg_outi("sid[2]=",bsid[2]);



        cp = radix64encode_noslash((char *) bsid, 12);
        if (cp == NULL) return TCL_ERROR;
        sprintf(temp,"/@@%s", cp);
        Tcl_SetResult(interp, temp, TCL_VOLATILE);
        if (cp != NULL) free(cp);
        return TCL_OK;

}
```

```
/* TclIdSid
 *    Scans an ascii line and finds an ascii SID. (no validation though)
 * Inputs:
 *    lineoftext
 * Returns:
 *    ascii bin_sid, if a sid is found it is returned.
 *
 */

int TclIdSid(ClientData dummy, Tcl_Interp *interp,
                          int argc, char **argv)
    {
    char *sidp, *cp;

    interp->result[0] = 0;


    if (argc != 2)
      {
      interp->result = "wrong # args";
      return TCL_ERROR;
      }
    sidp = (char *) strstr(argv[1], "/@@");
    if (sidp == NULL) return TCL_OK;
    cp = (char *) strstr(sidp+1,"/");
    if ((cp == NULL) && (strlen(sidp) != 19)) return TCL_OK;
    if ((cp - sidp) != 19) return TCL_OK;
    strncpy(interp->result, sidp,19);
    interp->result[19] = 0;
    return TCL_OK;
    }



/*
 * Register commands with interpreter.
 */
int SidSupInit(Tcl_Interp *interp)
  {
  Tcl_CreateCommand(interp, "packsid",   TclPackSid,   NULL, NULL);
  Tcl_CreateCommand(interp, "unpacksid", TclUnpackSid, NULL, NULL);
  Tcl_CreateCommand(interp, "unpacksidnovalidate", TclUnpackSidNoValidate, NULL,
  Tcl_CreateCommand(interp, "issid",     TclIdSid,     NULL, NULL);
  return TCL_OK;
  }
```

```
/*
 *----------------------------------------------------------------------
 *
 * compute_ihash --
 *
 *  Compute the MD5 hash for the specified string, returning the hash as
 *  a 32b xor of the 4 hash longwords.
 *
 * Results:
 *      hash int.
 *
 * Side effects:
 *      None.
 *----------------------------------------------------------------------
 */
int compute_ihash(char *str)
{
    MD5_CTX md5;
    unsigned char hash[16];
    unsigned int *p1;
    unsigned int hashi = 0;

    MD5Init(&md5);
    MD5Update(&md5, str, strlen(str));
    MD5Final(hash, &md5);
    p1 =  (unsigned int *) hash;

    hashi = *p1++;
    hashi ^= *p1++;
    hashi ^= *p1++;
    hashi ^= *p1++;
    return hashi;
}
```

```
/*
 * ticket.c --
 *
 *      Commands for TICKET.
 *
 * Copyright 1995 by Open Market, Inc.
 * All rights reserved.
 *
 * This file contains proprietary and confidential information and
 * remains the unpublished property of Open Market, Inc. Use,
 * disclosure, or reproduction is prohibited except as permitted by
 * express written license agreement with Open Market, Inc.
 *
 * Steve Morris
 * morris@OpenMarket.com
 *
 * Created: Wed Mar 1 1995
 * $Source: /omi/proj/master/omhttpd/Attic/ticket.c,v $
 *
 */

#if !defined(lint)
static const char rcsid[]="$Header: /omi/proj/master/omhttpd/Attic/ticket.c,v 2.
#endif /*not lint*/

#include <stdio.h>
#include <sys/utsname.h>
#include "httpd.h"
#include "md5.h"
#include "ticket.h"


static TICKET_Server TicketServerData;
```

43

```
/*
 * This file implements all the ticket/sid related functions for the server.
 *
 * The region commands RequireSID and xxxxx can be used to limit
 * access to groups of files based on the authentication of the requestor.
 * The two commands are very similar, and only differ in the method used to
 * present the authentication data (via the URL) and in handling of the
 * failing access case. For failing TICKET's, a "not authorized" message is
 * generated. For failing (or absent) SID's, a REDIRECT (either local or via
 * CGI script) is performed to forward the request to an authentication server.
 *
 * RequireSID domain1 [domain2 ... domainn]
 *
 *      This command denies access unless the specified properties are
 *      true of the request. Only one RequireSID or xxxxx command can
 *      be used for a given region, though it may specify multiple domains.
 *
 *
 */
```

43(a)

```
static int   ProcessRequires(ClientData clientData, Tcl_Interp *interp,
                             int argc, char **argv, int flavor);
static int   DomainNameCmd(ClientData clientData, Tcl_Interp *interp,
                             int argc, char **argv);
static int   GetDomain(char *domname, int dflt);
static char *GetAsciiDomain(char *domname, char *dflt);
static int   compute_ihash(char *str);
static char *computeHash(char *str);
static char *GetSecret(int kid);
static int   GetKidByKeyID(char *keyID);
static char *CreateSid(HTTP_Request *reqPtr, int dom, int uid, int kid,
                         int exp, int uctx);
static void freeTicketReqData(void *dataPtr);
static void DumpStatus(HTTP_Request *reqPtr);
static void TICKET_DebugHooks(ClientData clientData, char *suffix,
                            HTTP_Request *reqPtr);
static int ParseSid(HTTP_Request *reqPtr);
static int ParseTicket(HTTP_Request *reqPtr);
static char *fieldParse(char *str, char sep, char **endptr);
void TICKET_ConfigCheck();
void DumpRusage(HTTP_Request *reqPtr);
```

44

```
/*
 *----------------------------------------------------------------------
 *
 * TICKET_RequireSidCmd --
 *
 *      Checks that the requested URL is authorized via SID to access this
 *      region. If the access is not authorized and we do not have a "remote
 *      authentication server" registered, then an "unauthorized message"
 *      is returned. If a "remote authentication server" has been
 *      declared, we REDIRECT to that server, passing the requested URL and
 *      required domain's as arguments.
 *
 * Results:
 *      Normal Tcl result, or a REDIRECT request.
 *
 * Side effects:
 *      Either an "unauthorized access" message or a REDIRECT in case of error.
 *
 *----------------------------------------------------------------------
 */
static int TICKET_RequireSidCmd(ClientData clientData, Tcl_Interp *interp,
                            int argc, char **argv)
    {
    if (TicketGlobalData(EnableSidEater)) return TCL_OK;
    return(ProcessRequires(clientData, interp,argc, argv, ticketSid));
    }
```

45

```
/*
 *----------------------------------------------------------------------
 *
 * ProcessRequires --
 *
 *      Checks that the requested URL is authorized to access this
 *      region. The error cases are treated differently for SID v.s. TICKET.
 *      For Ticket's, an unauthorized access generates a returned error message.
 *      For SID's, we first look to see if we are operating in "local authentica
 *      mode", if we are, we generate a new SID, into the URL and re-process the
 *      If not in "local" mode, we look for the presence of a remoteauthenticati
 *      server, if we have one declared (in the conf file) we REDIRECT to it pas
 *      the FULL url and a list of domains that would have been legal. If the
 *      authentication server was not found we return an error message.
 *
 * Results:
 *      Normal Tcl result, a local reprocess command, or a REDIRECT request.
 *
 * Side effects:
 *      Either an "unauthorized access" message or a REDIRECT in case of error.
 *
 *----------------------------------------------------------------------
 */
static int ProcessRequires(ClientData clientData, Tcl_Interp *interp,
                           int argc, char **argv, int flavor)
    {
    HTTP_Request *reqPtr = (HTTP_Request *) clientData;
    HTTP_Server  *serverPtr;
    TICKET_Request *ticketPtr;
    DString targetUrl;
    DString escapeUrl;
    int i, required_dom;
    int firstLegalDom = -1;
    char *NewSid, *cp;

    DStringInit(&targetUrl);
    DStringInit(&escapeUrl);

    /* fetch the server private and ticket specific extension data */
    serverPtr = reqPtr->serverPtr;
    ticketPtr = (TICKET_Request *) HT_GetReqExtData(reqPtr, TicketServerData.tic
    ASSERT (ticketPtr != NULL);


    /* compare the requesting SID/Ticket<DOM> to authorized list of domains */
    /* a match OR any valid domain and a required domain of TicketFreeArea is su
    for (i = 1; i < argc; i++)
      {
      required_dom = GetDomain(argv[i],-1);
      if (required_dom != -1)
        {
        if (firstLegalDom == -1) firstLegalDom = required_dom;
        if ( (ticketPtr->sidDom == required_dom) ||
             (ticketPtr->valid && (ticketPtr->sidDom != -1) &&
              (required_dom == TicketGlobalData(FreeArea))) ||
             ((ticketPtr->ticketDom == required_dom) &&
               (time(0) <= ticketPtr->ticketExp) &&
               ((DStringLength(&ticketPtr->ticketIP) == 0) ||
```

46

```
          (strcmp(DStringValue(&ticketPtr->ticketIP), DStringValue(&reqPtr->r
        )
         {
         DStringFree(&targetUrl);
         DStringFree(&escapeUrl);
         return TCL_OK;
         }
      }
    }


  /* count the number of domain crossing that caused re-auth */
  if ((flavor == ticketSid) && (ticketPtr->sidDom) != -1) IncTicketCounter(Cou

  /* authorization failed, if this was a sid url, and local auth is enabled */
  /* or this was an access to the free area */
  /* insert a new sid in the url, and REDIRECT back to the client */
  if (TicketGlobalData(EnableLocalAuth) ||
      ((firstLegalDom == TicketGlobalData(FreeArea))
      && (flavor == ticketSid) && (firstLegalDom != -1)))
    {
    if ((DStringLength(&reqPtr->url) != 0) &&
        (DStringValue(&reqPtr->url)[0] != '/'))
      {
      HTTP_Error(reqPtr, NOT_FOUND, "access denied due to poorly formed url");
      DStringFree(&targetUrl);
      DStringFree(&escapeUrl);
      if (!ticketPtr->valid)
        DStringFree(&ticketPtr->sid);
      return TCL_RETURN ;
      }
    NewSid =  CreateSid(reqPtr,
              firstLegalDom, ticketPtr->uid,
              TicketGlobalData(CurrentSecret), TicketGlobalData(LocalAuthExp),
              ticketPtr->uctx);
    DStringFree(&ticketPtr->sid);
    DStringAppend(&ticketPtr->sid, NewSid, -1);
    ComposeURL(reqPtr, DStringValue(&reqPtr->url), &targetUrl);
    IncTicketCounter(CountLocalRedirects);
    HTTP_Error(reqPtr, REDIRECT, DStringValue(&targetUrl));
    DStringFree(&targetUrl);
    DStringFree(&escapeUrl);
    if (!ticketPtr->valid)
      DStringFree(&ticketPtr->sid);
    return TCL_RETURN;
    }


  /* authorization failed, build the REDIRECT URL arg's. */
  /* If present, REDIRECT to authentication server */
  if ((DStringLength(&TicketGlobalData(AuthServer)) != 0) && (flavor == ticket
    {
    if ((DStringLength(&reqPtr->url) != 0) &&
        (DStringValue(&reqPtr->url)[0] != '/'))
      {
      HTTP_Error(reqPtr, NOT_FOUND, "access denied due to poorly formed url");
      DStringFree(&targetUrl);
      DStringFree(&escapeUrl);
      if (!ticketPtr->valid)
        DStringFree(&ticketPtr->sid);
    return TCL_RETURN ;
    }
  DStringAppend(&targetUrl, DStringValue(&TicketGlobalData(AuthServer)), -1)
  DStringAppend(&targetUrl, "?url=", -1);
  ComposeURL(reqPtr, DStringValue(&reqPtr->url), &escapeUrl);
  EscapeUrl(&escapeUrl);
  DStringAppend(&targetUrl,DStringValue(&escapeUrl), -1);
  DStringAppend(&targetUrl, "&domain=", -1);
  DStringTrunc(&escapeUrl, 0);
  DStringAppend(&escapeUrl, "{", -1);
  for (i=1; i < argc; i++)
    {
    cp = GetAsciiDomain(argv[i], NULL);
    if (cp != NULL)
      {
      DStringAppend(&escapeUrl, cp, -1);
      DStringAppend(&escapeUrl, " ", -1);
      }
    }
  DStringAppend(&escapeUrl, "}", -1);
  EscapeUrl(&escapeUrl);
  DStringAppend(&targetUrl,DStringValue(&escapeUrl), -1);
  DStringFree(&escapeUrl);
  HTTP_Error(reqPtr, REDIRECT, DStringValue(&targetUrl));
  IncTicketCounter(CountRemoteRedirects);
  DStringFree(&targetUrl);
  if (!ticketPtr->valid)
    DStringFree(&ticketPtr->sid);
  return TCL_RETURN;
  }

/* authorization failed, if this is a ticket access, decode the */
/* reason and handle via a redirect to a handler, or punt a    */
/* no access message */
if ((flavor == ticketTicket) && (firstLegalDom != -1) && (ticketPtr->ticketD
  {
  /* check For IP address restrictions */
  if ((DStringLength(&ticketPtr->ticketIP) != 0) &&
      (DStringLength(&TicketGlobalData(TicketAdrHandler)) != 0) &&
      (strcmp(DStringValue(&ticketPtr->ticketIP), DStringValue(&reqPtr->remo
    {
    DStringAppend(&targetUrl, DStringValue(&TicketGlobalData(TicketAdrHandle
    DStringAppend(&targetUrl, DStringValue(&ticketPtr->fields), -1);
    DStringAppend(&targetUrl, "&url=", -1);
    DStringAppend(&targetUrl, DStringValue(&reqPtr->url), -1);
    IncTicketCounter(CountTicketAddr);
    HTTP_Error(reqPtr, REDIRECT, DStringValue(&targetUrl));
    DStringFree(&targetUrl);
    return TCL_RETURN;
    }

  /* check for expired tickets */
  if (time(0) > ticketPtr->ticketExp)
    {
    DStringAppend(&targetUrl, DStringValue(&TicketGlobalData(TicketExpHandle
    DStringAppend(&targetUrl, DStringValue(&ticketPtr->fields), -1);
    DStringAppend(&targetUrl, "&url=", -1);
    DStringAppend(&targetUrl, DStringValue(&reqPtr->url), -1);
    IncTicketCounter(CountExpiredTicket);
```

48 -

```
    HTTP_Error(reqPtr, REDIRECT, DStringValue(&targetUrl));
    DStringFree(&targetUrl);
    return TCL_RETURN;
    }
  }

/* no handler, punt a message */
HTTP_Error(reqPtr, FORBIDDEN, "access denied by Require ticket/sid region co
IncTicketCounter(CountNoRedirects);
if (!ticketPtr->valid)
DStringFree(&ticketPtr->sid);
DStringFree(&targetUrl);
DStringFree(&escapeUrl);
return TCL_RETURN;
}
```

```
/*
 *----------------------------------------------------------------------
 *
 * Get[Ascii]Domain --
 *
 *      These routine performs an ascii to binary domain name lookup,
 *      indexed by 'key') from the server's domain name catalog.  Name/number
 *      pair's are loaded into the catalog at configuration time with the
 *      with the "Domain" configuration command. The Ascii version returns
 *      a pointer to a character string that represents the domain number.
 *      The non Ascii version returns an integer representing the domain number.
 *
 * Results:
 *      Integer value of domain.  If no domain is available, returns deflt.
 *
 * Side effects:
 *      None.
 *
 *----------------------------------------------------------------------
 */
static int GetDomain(char *domname, int deflt)
    {
    HashEntry *entryPtr;
    DString DomName;


    DStringInit(&DomName);
    DStringAppend(&DomName, domname, -1);
    strtolower(DStringValue(&DomName));


    entryPtr = FindHashEntry(&TicketServerData.Domains, DStringValue(&DomName));
    DStringFree(&DomName);
    if (entryPtr == NULL) return deflt;
    return (int) GetHashValue(entryPtr);
    }

static char * GetAsciiDomain(char *domname, char *deflt)
    {
    HashEntry *entryPtr;
    static char buffer[64];
    DString DomName;


    DStringInit(&DomName);
    DStringAppend(&DomName, domname, -1);
    strtolower(DStringValue(&DomName));

    entryPtr = FindHashEntry(&TicketServerData.Domains, DStringValue(&DomName));
    DStringFree(&DomName);
    if (entryPtr == NULL) return deflt;
    sprintf(buffer,"%d", (int) GetHashValue(entryPtr));
    return buffer;
   }
```

50

```
/*
 *----------------------------------------------------------------------
 *
 * TICKET_InsertLocalSid --
 *
 *      Given a URL, inspect it to see if it refers to the local server/port
 *      if it does, and it does not already contain a SID, insert one if
 *      the current request included one. Note, for port 80 access we look
 *      for a match with and without the port specifier.
 *
 * Results:
 *      None.
 *
 * Side effects:
 *      A SID may be inserted into the URL.
 *
 *----------------------------------------------------------------------
 */
void TICKET_InsertLocalSid(HTTP_Request *reqPtr, DString *result)
    {
    HTTP_Server *serverPtr;
    TICKET_Request *ticketPtr;
    char tmp[32];
    DString pattern1;
    DString pattern2;
    DString tmp_url;
    DString *hitPattern = NULL;


    ticketPtr = (TICKET_Request *) HT_GetReqExtData(reqPtr, TicketServerData.tic
    if (ticketPtr == NULL) return;
    serverPtr = reqPtr->serverPtr;

    DStringInit(&pattern1);
    DStringInit(&pattern2);
    DStringInit(&tmp_url);


    DStringAppend(&pattern1, "http://", -1);
    DStringAppend(&pattern1, DStringValue(&serverPtr->serverName), -1);
    DStringAppend(&pattern2, DStringValue(&pattern1), -1);
    sprintf(tmp, ":%d", serverPtr->server_port);
    DStringAppend(&pattern1, tmp, -1);

    if ((DStringLength(result) >= DStringLength(&pattern1)) &&
        (strncasecmp(DStringValue(&pattern1), DStringValue(result), DStringLengt
        hitPattern = &pattern1;
    else
    if ((serverPtr->server_port == 80) &&
        (DStringLength(result) >= DStringLength(&pattern2)) &&
        (strncasecmp(DStringValue(&pattern2), DStringValue(result), DStringLengt
        hitPattern = &pattern2;

    if (hitPattern != NULL)
      {
      DStringAppend(&tmp_url, DStringValue(hitPattern), -1);
      DStringAppend(&tmp_url, DStringValue(&ticketPtr->sid), -1);
      DStringAppend(&tmp_url, &DStringValue(result)[DStringLength(hitPattern)],
      DStringFree(result);
```

51

```
  DStringAppend(result, DStringValue(&tmp_url), -1);
  DStringFree(&tmp_url);
  }
DStringFree(&pattern1);
DStringFree(&pattern2);
DStringFree(&tmp_url);
}
```

52

```
/*
 *----------------------------------------------------------------------
 *
 * CreateSid --
 *
 *      This routine takes the passed arguments and creates a sid.
 *
 * Results:
 *      A sid.
 *
 * Side effects:
 *
 *----------------------------------------------------------------------
 */

char * CreateSid(HTTP_Request *reqPtr, int dom, int uid, int kid,
                        int exp, int uctx)

    {
    int bsid[3]= {0,0,0};
    char temp_str[512];
    DString hash;
    int act_hash;
    static char sid[64];
    unsigned int expire_time;
    char *secret;
    char *hashP;
    char *cp;
    unsigned char *ecp;
    unsigned int eda;
    int endian = 1;



    DStringInit(&hash);
    expire_time = time(0)+ exp;

    put_sid(dom_lw,   dom_pos,   dom_mask,   dom);
    put_sid(uid_lw,   uid_pos,   uid_mask,   uid);
    put_sid(kid_lw,   kid_pos,   kid_mask,   kid);
    put_sid(exp_lw,   exp_pos,   exp_mask,   (expire_time>>exp_shft_amt))
    put_sid(uctx_lw,  uctx_pos,  uctx_mask,  uctx);
    put_sid(rev_lw,   rev_pos,   rev_mask,   sid_rev_zero);

    secret = GetSecret(kid);
    ASSERT (secret != NULL);
    DStringAppend(&hash, secret, -1);
    DStringAppend(&hash, DStringValue(&reqPtr->remoteAddr), -1);
    sprintf(temp_str, "%08x%08x", bsid[2],bsid[1]);
    DStringAppend(&hash, temp_str, -1);
    /* format of the hash string is %s%s%08x%08x", secret,ip_addr,bsid[2],bsid[1

    hashP = DStringValue(&hash);
    act_hash = compute_ihash(hashP);
    while (*hashP != 0) *hashP++ = 0;
    DStringFree(&hash);

/*  fix_endian(&act_hash, ecp, eda); */
```

53

```
   .put_sid(sig_lw, sig_pos, sig_mask, act_hash)

/*  fix_endian(&bsid[0], ecp, eda); */
    fix_endian(&bsid[1], ecp, eda);
    fix_endian(&bsid[2], ecp, eda);

#if (1 == 0)
    DumpSid();
#endif

    cp = radix64encode_noslash((char *) bsid, 12);
    strcpy(sid, SID_prefix);
    strcat(sid, cp);
    free(cp);
    return(sid);
    }
```

54

```
/*
 *----------------------------------------------------------------------
 *
 * compute_ihash --
 *
 *  Compute the MD5 hash for the specified string, returning the hash as
 *  a 32b xor of the 4 hash longwords.
 *
 * Results:
 *      hash int.
 *
 * Side effects:
 *      None.
 *----------------------------------------------------------------------
 */
static int compute_ihash(char *str)
    {
    MD5_CTX md5;
    unsigned char hash[16];
    unsigned int *p1;
    unsigned int hashi = 0;

    MD5Init(&md5);
    MD5Update(&md5, (unsigned char *) str, strlen(str));
    MD5Final(hash, &md5);
    p1 =  (unsigned int *) hash;

    hashi = *p1++;
    hashi ^= *p1++;
    hashi ^= *p1++;
    hashi ^= *p1++;
    return hashi;
    }
```

55

```
/*
 *----------------------------------------------------------------------
 *
 * computeHash --
 *
 *  Compute the MD5 hash for the specified string, returning the hash as
 *  a 32-character hex string.
 *
 * Results:
 *      Pointer to static hash string.
 *
 * Side effects:
 *      None.
 *----------------------------------------------------------------------
 */
static char *computeHash(char *str)
{
    int i;
    MD5_CTX md5;
    unsigned char hash[16];
    static char hashstr[33];
    char *q;

    MD5Init(&md5);
    MD5Update(&md5, (unsigned char *) str, strlen(str));
    MD5Final(hash, &md5);
    q = hashstr;
    for(i=0; i<16; i++) {
        sprintf(q, "%02x", hash[i]);
        q += 2;
    }
    *q = '\0';
    return hashstr;
}
```

56

```
/*
 *----------------------------------------------------------------------
 *
 * TICKET_ParseTicket --
 *
 *      Called by dorequest, before any region commands or mount handlers
 *      have run. We parse and handle incomeing sid's and tickets.
 *
 * Results:
 *      None.
 *
 * Side effects:
 *
 *----------------------------------------------------------------------
 */

int TICKET_ParseTicket(HTTP_Request *reqPtr)
    {
    int status = HT_OK;

    IncTicketCounter(CountTotalUrl);

    status = ParseSid(reqPtr);
    if (TicketGlobalData(EnableTicket) && (status == HT_OK)) status = ParseTicke
    return status;
    }
```

57

```
/*
 *----------------------------------------------------------------------
 *
 * ParseSid --
 *
 *      Called by TICKET_ParseTicket, before any region commands or mount handle
 *      have run. We parse and handle incomeing sid's.
 *
 * Results:
 *      None.
 *
 * Side effects:
 *
 *----------------------------------------------------------------------
 */

int ParseSid(HTTP_Request *reqPtr)
    {
    TICKET_Request *ticketPtr;
    HTTP_Server *serverPtr;
    DString hash;
    int i;
    char *cp, *cp1;
    int *bsid=NULL, act_hash;
    unsigned int cur_tim, tdif, exp_tim;
    char *secret;
    char temp_str[512];
    char *hashP;
    int sid_ok = 0;
    unsigned char *ecp;
    unsigned int eda;
    int endian = 1;
    int ip1,ip2,ip3,ip4;


    /* fetch the server private ticket extension data */
    /* note that this sets up a default ticket block for both SID's and Ticket a
    serverPtr = reqPtr->serverPtr;
    ticketPtr = (TICKET_Request *) HT_GetReqExtData(reqPtr, TicketServerData.tic
    ASSERT (ticketPtr == NULL);

    ticketPtr = (TICKET_Request *) Malloc(sizeof(TICKET_Request));
    HT_AddReqExtData(reqPtr, TicketServerData.ticketExtensionId, ticketPtr, free
    DStringInit(&ticketPtr->rawUrl);
    DStringInit(&ticketPtr->sid);
    DStringInit(&ticketPtr->fields);
    DStringInit(&ticketPtr->signature);
    DStringInit(&ticketPtr->ticketIP);
    ticketPtr->valid     = 0;
    ticketPtr->sidDom    = -1;
    ticketPtr->ticketDom = -1;
    ticketPtr->ticketExp = -1;
    ticketPtr->uid       = 0;
    ticketPtr->uctx      = 0;
    sscanf(DStringValue(&reqPtr->remoteAddr), "%d.%d.%d.%d", &ip1, &ip2, &ip3, &
    ticketPtr->uid = (((ip1+ip2)<<24) | ((ip3+ip4)<<16) | (rand() & 0xFFFF));
    ticketPtr->uctx  = 1;
```

58

```
/* we are done if sids are not enabled, or this url does not have a sid */
if (!(TicketGlobalData(EnableSid))) return HT_OK;
cp1 = DStringValue(&reqPtr->url);
if (strstr(cp1, SID_prefix) != cp1)
  return HT_OK;
if (strlen(cp1) == sidLength)
  {
  DStringAppend(&reqPtr->url, "/", -1);
  DStringAppend(&reqPtr->path, "/", -1);
  cp1 = DStringValue(&reqPtr->url);
  }
cp = strchr(cp1+sizeof(SID_prefix),'/');
if ((cp - cp1) != sidLength)
  return HT_OK;
IncTicketCounter(CountSidUrl);


DStringInit(&hash);

/* if sid eater is enabled, rewrite the url without the sid, and reprocess t
if (TicketGlobalData(EnableSidEater))
  {
  DStringAppend(&hash, DStringValue(&reqPtr->url), -1);
  DStringFree(&reqPtr->url);
  DStringAppend(&reqPtr->url, DStringValue(&hash)+sidLength, -1);
  DStringTrunc(&hash, 0);
  DStringAppend(&hash, DStringValue(&reqPtr->path), -1);
  DStringFree(&reqPtr->path);
  DStringAppend(&reqPtr->path, DStringValue(&hash)+sidLength, -1);
  DStringFree(&hash);
  IncTicketCounter(CountDiscardedSidUrl);
  return HT_OK;
  }


DStringAppend(&ticketPtr->sid,  DStringValue(&reqPtr->url), sidLength);

/* first convert the SID back to binary*/
i = DStringLength(&ticketPtr->sid)-3;
bsid = (int *) radix64decode_noslash(DStringValue(&ticketPtr->sid)+3, i, &i)
if ((bsid == NULL) || (i != 12)) goto rtn_exit;

fix_endian(&bsid[0], ecp, eda);
fix_endian(&bsid[1], ecp, eda);
fix_endian(&bsid[2], ecp, eda);

/* check the SID version field */
if (get_sid(rev_lw,rev_pos,rev_mask) != sid_rev_zero) goto sid_bad;
if (get_sid(rsrv1_lw,rsrv1_pos,rsrv1_mask) != 0) goto sid_bad;
if (get_sid(rsrv2_lw,rsrv2_pos,rsrv2_mask) != 0) goto sid_bad;

/* Get a pointer to the secret */
secret = GetSecret(get_sid(kid_lw,kid_pos,kid_mask));
if (secret == NULL) goto sid_bad;

 /* hash the sid and check the signature */
DStringAppend(&hash, secret, -1);
```

57

```
.DStringAppend(&hash, DStringValue(&reqPtr->remoteAddr), -1);
 sprintf(temp_str, "%08x%08x", bsid[2],bsid[1]);
 DStringAppend(&hash, temp_str, -1);
 /* format of the hash string is %s%s%08x%08x", secret,ip_addr,bsid[2],bsid[1

 hashP = DStringValue(&hash);
 act_hash = compute_ihash(hashP);
 while (*hashP != 0) *hashP++ = 0;


 fix_endian(&act_hash, ecp, eda);
 if (act_hash != get_sid(sig_lw,sig_pos,sig_mask)) goto  sid_bad;


 /* is is ok, may be expired, but good enough to id user */
 ticketPtr->uid = get_sid(uid_lw,uid_pos,uid_mask);
 ticketPtr->uctx = get_sid(uctx_lw,uctx_pos,uctx_mask);

 /* do the SID experation processing */
 cur_tim = (time(0)>>exp_shft_amt) & exp_mask;
 exp_tim = get_sid(exp_lw,exp_pos,exp_mask);
 tdif = (exp_tim - cur_tim) & 0xffff;
 if (tdif > 0x7fff)
   {
   IncTicketCounter(CountExpSid);
   goto sid_exp;
   }

 /* sid is fine, save the sid state, update the url's */
 ticketPtr->sidDom = get_sid(dom_lw,dom_pos,dom_mask);
 ticketPtr->valid = 1;
 sid_ok = 1;
 IncTicketCounter(CountValidSid);

sid_bad:
  if (!(sid_ok)) IncTicketCounter(CountInvalidSid);
sid_exp:
  DStringAppend(&ticketPtr->rawUrl, DStringValue(&reqPtr->path), -1);
  DStringTrunc(&reqPtr->path, 0);
  DStringAppend(&reqPtr->path, DStringValue(&ticketPtr->rawUrl)+sidLength, -1)

  DStringTrunc(&ticketPtr->rawUrl, 0);
  DStringAppend(&ticketPtr->rawUrl, DStringValue(&reqPtr->url), -1);
  DStringTrunc(&reqPtr->url, 0);
  DStringAppend(&reqPtr->url, DStringValue(&ticketPtr->rawUrl)+sidLength, -1);

rtn_exit:
  DStringFree(&hash);
  if (bsid != NULL) free(bsid);
  return HT_OK;
  }
```

60

```
/*
 *------------------------------------------------------------------
 *
 * freeTicketReqData
 *
 *       This routine frees the storage used by ticket specific request
 *       data.
 *
 * Results:
 *      None.
 *
 * Side effects:
 *      Memory freed.
 *
 *------------------------------------------------------------------
 */

static void freeTicketReqData(void *dataPtr)
    {
    TICKET_Request *ticketPtr = dataPtr;
    DStringFree(&ticketPtr->rawUrl);
    DStringFree(&ticketPtr->sid);
    DStringFree(&ticketPtr->fields);
    DStringFree(&ticketPtr->signature);
    DStringFree(&ticketPtr->ticketIP);
    free(ticketPtr);
    }
```

- 61 -

```
/*
 *----------------------------------------------------------------------
 *
 * GetSecret --
 *
 *      Given a binary keyID, returns an ascii secret from the
 *      secrets store.
 *      for untranslatable names, return NULL.
 *
 * Results:
 *      "I've got a secret, now you do too"
 *
 * Side effects:
 *
 *
 *----------------------------------------------------------------------
 */

char *GetSecret(int kid)
    {
    HashEntry *entryPtr;

    entryPtr = FindHashEntry(&TicketServerData.SecretsKid, (void *) kid);
    if(entryPtr == NULL) return NULL;
    return DStringValue((DString *)GetHashValue(entryPtr));
    }
```

```
/*
 *-------------------------------------------------------------------------
 *
 * GetKidByKeyID --
 *
 *      Given an ascii  KeyID return the binary KeyID.
 *      for untranslatable names, return -1.
 *
 * Results:
 *      "I've got a secret, now you do too"
 *
 * Side effects:
 *
 *
 *-------------------------------------------------------------------------
 */

int GetKidByKeyID(char *keyID)
    {
    HashEntry *entryPtr;

    entryPtr = FindHashEntry(&TicketServerData.KeyID, (void *) keyID);
    if(entryPtr == NULL) return -1;
    return (int) GetHashValue(entryPtr);
    }
```

```
/*
 *----------------------------------------------------------------------
 *
 * fieldParse --
 *
 *  Given a string, a separator character, extracts a field up to the
 *  separator into the result string.
 *  Does substitution on '%XX' sequences, and returns the pointer to the
 *  character beyond last character in '*endptr'.
 *
 * Results:
 *      Returns a malloc'ed string (caller must free), or NULL if an
 *      error occurred during processing (such as an invalid '%' sequence).
 *
 * Side effects:
 *      None.
 *
 *----------------------------------------------------------------------
 */
#define SIZE_INC 200
static char *fieldParse(char *str, char sep, char **endptr)
{
    char buf[3];
    char c;
    char *end, *data, *p;
    int maxlen, len;

    len = 0;
    maxlen = SIZE_INC;
    p = data = malloc(maxlen);

  /*
   * Loop through string, until end of string or sep character.
   */
    while(*str && *str != sep) {

        if(*str == '%') {
            if(!isxdigit(str[1]) || !isxdigit(str[2])) {
                free(data);
                return NULL;
            }
            buf[0] = str[1];
            buf[1] = str[2];
            buf[2] = '\0';
            c = strtol(buf, &end, 16);
            str += 3;
        } else if(*str == '+') {
            c = ' ';
            str++;
        } else
            c = *str++;

        *p++ = c;
        len++;
        if(len >= maxlen) {
            maxlen += SIZE_INC;
            data = realloc(data, maxlen);
            p = data + len;
        }
```

64

```
    }
    *p++ = '\0';
    *endptr = str;
    return data;
}
```

65

```
/*
 *----------------------------------------------------------------------
 *
 * DomainNameCmd --
 *
 *      A call to this routine, builds the ascii domain name
 *      to binary domain name maping structure for a numeric domain.
 *      Syntax is Domain number name1 name2 name3 name... name_last
 *      At least one name is required. The number is decimal and
 *      can be any value except -1. -1 is reserved as a marker
 *      for untranslatable names.
 *
 * Results:
 *      None.
 *
 * Side effects:
 *      Commands are validate, and entries added to the map
 *
 *----------------------------------------------------------------------
 */

static int DomainNameCmd(ClientData clientData, Tcl_Interp *interp,
                         int argc, char **argv)
    {
    int new,i;
    HashEntry *entryPtr;
    int DomNumber;
    DString DomName;


    if (argc <3)
      {
      Tcl_AppendResult(interp, argv[0], " directive:  wrong number of "
                "arguments, should be \"3\"",
                (char *) NULL);
      return TCL_ERROR;
      }

    DStringInit(&DomName);


    if (((sscanf(argv[1], "%d", &DomNumber) != 1) || (DomNumber == -1)))
      {
      Tcl_AppendResult(interp, argv[0], " directive:  ",
                "Domain number must be an integer, and not equal to -1",
                ", value found was ",argv[1],
                (char *) NULL);
      return TCL_ERROR;
      }

    for (i = 2; i < argc; i++)
      {
      DStringFree(&DomName);
      DStringAppend(&DomName, argv[i], -1);
      strtolower(DStringValue(&DomName));
      entryPtr = CreateHashEntry(&TicketServerData.Domains, DStringValue(&DomNam
      if (new == 0)
        {
        Tcl_AppendResult(interp, argv[0], " directive:  ",
```

66

```
                "Duplicate domain name specified, '", argv[i], "'",
                (char *) NULL);
        return TCL_ERROR;
        }
      SetHashValue(entryPtr, DomNumber);
      }
    DStringFree(&DomName);
    return TCL_OK;
    }
```

67

```
/*
 *-----------------------------------------------------------------------
 *
 * SecretsCmd --
 *
 *      A call to this routine, builds kid to secrets table
 *
 * Results:
 *      None.
 *
 * Side effects:
 *      Secrets are stored.
 *
 *-----------------------------------------------------------------------
 */

static int SecretsCmd(ClientData clientData, Tcl_Interp *interp,
                          int argc, char **argv)
    {
    int newKid,newKeyID;
    HashEntry *entryPtrKid = NULL , *entryPtrKeyID = NULL;
    int Kid;
    DString *dsptrKid;


    if (argc != 4)
      {
      Tcl_AppendResult(interp, argv[0], " directive:  wrong number of "
                "arguments, should be \"4\"",
                (char *) NULL);
      return TCL_ERROR;
      }

    if (sscanf(argv[2], "%d", &Kid) != 1)
      {
      Tcl_AppendResult(interp, argv[0],
                " directive:  KeyID must be an integer",
                ", value found was '",argv[2],"'",
                (char *) NULL);
      return TCL_ERROR;
      }

    entryPtrKid = CreateHashEntry(&TicketServerData.SecretsKid, (void *) Kid, &n
    if (strlen(argv[1]))
      entryPtrKeyID = CreateHashEntry(&TicketServerData.KeyID, (void *) argv[1],
    if ((newKid == 0) || ((newKeyID == 0) && strlen(argv[1])))
      {
      Tcl_AppendResult(interp, argv[0],
                " directive:  Duplicate Secret specified for KeyID '",
                argv[1],
                (char *) NULL);
      return TCL_ERROR;
      }

    if (strlen(argv[1]))
      {
      dsptrKid = (DString *) malloc(sizeof(DString));
      DStringInit(dsptrKid);
      DStringAppend(dsptrKid, argv[3], -1);
```

68

```
  SetHashValue(entryPtrKid, dsptrKid);
  }
SetHashValue(entryPtrKeyID, Kid);
return TCL_OK;
}
```

```
/*
 *----------------------------------------------------------------------
 *
 * TICKET_Initialize --
 *
 *      Calls all the necessary routines to initialize the ticket subsystem.
 *
 * Results:
 *      None.
 *
 * Side effects:
 *      Commands added to the region interpreter.
 *      SID "/@@" url catcher declared.
 *
 *----------------------------------------------------------------------
 */

int TICKET_Initialize(HTTP_Server *serverPtr, Tcl_Interp *interp)
    {
    TicketServerData.ticketExtensionId = HT_RegisterExtension(serverPtr, "ticket

    InitHashTable(&TicketServerData.SecretsKid, TCL_ONE_WORD_KEYS);
    InitHashTable(&TicketServerData.KeyID,     TCL_STRING_KEYS);
    InitHashTable(&TicketServerData.Domains,  TCL_STRING_KEYS);

    /* initialize Server ticket data */
    DStringInit(&TicketGlobalData(AuthServer));
    DStringInit(&TicketGlobalData(TicketExpHandler));
    DStringInit(&TicketGlobalData(TicketAdrHandler));
    TicketGlobalData(FreeArea)              = 0;
    TicketGlobalData(EnableLocalAuth)       = 0;
    TicketGlobalData(CurrentSecret)         = 0;
    TicketGlobalData(EnableSid)             = 0;
    TicketGlobalData(EnableTicket)          = 0;
    TicketGlobalData(EnableSidEater)        = 0;
    TicketGlobalData(LocalAuthExp)          = 60*30;

    /* ticket event counters */
    TicketGlobalData(CountTotalUrl)         = 0;
    TicketGlobalData(CountSidUrl)           = 0;
    TicketGlobalData(CountValidSid)         = 0;
    TicketGlobalData(CountExpSid)           = 0;
    TicketGlobalData(CountInvalidSid)       = 0;
    TicketGlobalData(CountCrossDomain)      = 0;
    TicketGlobalData(CountLocalRedirects)   = 0;
    TicketGlobalData(CountRemoteRedirects)  = 0;
    TicketGlobalData(CountNoRedirects)      = 0;
    TicketGlobalData(CountDiscardedSidUrl)  = 0;


    /* Ticket related Config commands */
    Tcl_CreateCommand(interp, "Domain",                  DomainNameCmd,
                      (ClientData) serverPtr, NULL);
    Tcl_CreateCommand(interp, "Secrets",                 SecretsCmd,
                      (ClientData) serverPtr, NULL);
    Tcl_CreateCommand(interp, "AuthenticationServer", CmdStringValue,
            (ClientData) &TicketGlobalData(AuthServer), NULL);
    Tcl_CreateCommand(interp, "TicketExpirationHandler", CmdStringValue,
            (ClientData) &TicketGlobalData(TicketExpHandler), NULL);
```

70

```
Tcl_CreateCommand(interp, "TicketAddressHandler", C…dStringValue,
          (ClientData) &TicketGlobalData(TicketAdrHandler), NULL);
Tcl_CreateCommand(interp, "FreeDomain",           CmdIntValue,
          (ClientData) &TicketGlobalData(FreeArea), NULL);
Tcl_CreateCommand(interp, "EnableSidEater",       CmdIntValue,
          (ClientData) &TicketGlobalData(EnableSidEater), NULL);
Tcl_CreateCommand(interp, "EnableSid",            CmdIntValue,
          (ClientData) &TicketGlobalData(EnableSid), NULL);
Tcl_CreateCommand(interp, "EnableTicket",            CmdIntValue,
          (ClientData) &TicketGlobalData(EnableTicket), NULL);
Tcl_CreateCommand(interp, "EnableLocalAuth",      CmdIntValue,
          (ClientData) &TicketGlobalData(EnableLocalAuth), NULL);
Tcl_CreateCommand(interp, "CurrentSecret",        CmdIntValue,
          (ClientData) &TicketGlobalData(CurrentSecret), NULL);
Tcl_CreateCommand(interp, "LocalAuthExp",         CmdIntValue,
          (ClientData) &TicketGlobalData(LocalAuthExp), NULL);

HT_AddMountHandler(serverPtr, (ClientData) NULL, TICKET_DebugHooks,
    "/omiserver", NULL);

return HT_OK;
}
```

- 71 -

```
/*
 *----------------------------------------------------------------------
 *
 * TICKET_Shutdown --
 *
 *      Calls all the necessary routines to shutdown the ticket subsystem.
 *
 * Results:
 *      None.
 *
 * Side effects:
 *      Memory freed
 *
 *----------------------------------------------------------------------
 */

void TICKET_Shutdown(HTTP_Server *serverPtr)
    {
    HashEntry *entryPtr;
    HashSearch search;
    DString   *dstring;

    DStringFree(&TicketGlobalData(AuthServer));
    DStringFree(&TicketGlobalData(TicketExpHandler));
    DStringFree(&TicketGlobalData(TicketAdrHandler));

    entryPtr = FirstHashEntry(&TicketServerData.SecretsKid, &search);
    while (entryPtr != NULL)
      {
      dstring = GetHashValue(entryPtr);
      DStringFree(dstring);
      free(dstring);
      entryPtr = NextHashEntry(&search);
      }
    DeleteHashTable(&TicketServerData.SecretsKid);

    DeleteHashTable(&TicketServerData.KeyID);

    DeleteHashTable(&TicketServerData.Domains);
    }
```

72

```
/*
 *----------------------------------------------------------------------
 *
 * TICKET_AddRegionCommands --
 *
 *      Add TICKET region commands for authentication/authorization decisions.
 *
 * Results:
 *      None.
 *
 * Side effects:
 *      Commands added to the region interpreter.
 *
 *----------------------------------------------------------------------
 */

void TICKET_AddRegionCommands(HTTP_Request *reqPtr, Tcl_Interp *interp)
    {
    Tcl_CreateCommand(interp, "RequireSID", TICKET_RequireSidCmd,
                    (ClientData) reqPtr, NULL);
    Tcl_CreateCommand(interp, "RequireTicket", TICKET_RequireTicketCmd,
                      (ClientData) reqPtr, NULL);
    }
```

73

```
/*
 *----------------------------------------------------------------
 *
 * TICKET_GetCGIVariables --
 *
 *      Add TICKET CGI variables to the CGI variable table.
 *
 * Results:
 *      None.
 *
 * Side effects:
 *      Extends the CGI variable hash table.
 *
 *----------------------------------------------------------------
 */

void TICKET_GetCGIVariables(HTTP_Request *req)
    {
    TICKET_Request *ticketPtr = (TICKET_Request *) HT_GetReqExtData(req, TicketS


    /*
     * If there's no extension data, then we're not doing a ticket.  Just return
     */

    if (ticketPtr == NULL)
        return;

    if (DStringLength(&ticketPtr->rawUrl) != 0)
      HT_AddCGIParameter(req, "TICKET_URL", DStringValue(&ticketPtr->rawUrl), FA
    if (DStringLength(&ticketPtr->sid) != 0)
      HT_AddCGIParameter(req, "TICKET_SID", DStringValue(&ticketPtr->sid), FALSE
    if (DStringLength(&ticketPtr->fields) != 0)
      HT_AddCGIParameter(req, "TICKET_FIELDS", DStringValue(&ticketPtr->fields),
    if (DStringLength(&ticketPtr->signature) != 0)
      HT_AddCGIParameter(req, "TICKET_SIGNATURE", DStringValue(&ticketPtr->signa
    }
```

74

```
/*
 *----------------------------------------------------------------------
 *
 * TICKET_GetUrl
 *
 *      Return the original url (with sid)
 *
 * Results:
 *      The URL.
 *
 * Side effects:
 *      None.
 *
 *----------------------------------------------------------------------
 */
char * TICKET_GetUrl(HTTP_Request *reqPtr)
    {

    TICKET_Request *ticketPtr;

    ticketPtr = (TICKET_Request *)
      HT_GetReqExtData(reqPtr, TicketServerData.ticketExtensionId);
    if ((ticketPtr != NULL) &&
        (DStringLength(&ticketPtr->rawUrl) != 0))
      return DStringValue(&ticketPtr->rawUrl);
    else
      return DStringValue(&reqPtr->url);
    }
```

75

```
/*
 *-------------------------------------------------------------------
 *
 * TICKET_ConfigCheck
 *
 *      Perform late configuration checks
 *
 * Results:
 *
 *
 * Side effects:
 *      Possible message loged/printed, and program exit'd.
 *
 *-------------------------------------------------------------------
 */
void TICKET_ConfigCheck()
   {
   HashEntry *entryPtr;
   int kid;

   if ((TicketGlobalData(EnableSid) & ~0x1) != 0)
     {
     LogMessage(LOG_ERR, "EnableSid must be 0 or 1");
     exit(0);
     }

   if (!(TicketGlobalData(EnableSid))) return;

   kid = TicketGlobalData(CurrentSecret);
   if ((kid && kid_mask) != kid)
     {
     LogMessage(LOG_ERR, "CurrentSecret %d is invalid", kid);
     exit(0);
     }

   entryPtr = FindHashEntry(&TicketServerData.SecretsKid, (void *) kid);
   if(entryPtr == NULL)
     {
     LogMessage(LOG_ERR, "No secret defined for CurrentSecret %d", kid);
     exit(0);
     }

   if ((TicketGlobalData(FreeArea) & ~0x255) != 0)
     {
     LogMessage(LOG_ERR, "FreeArea must be between 0 and 255");
     exit(0);
     }

   if ((TicketGlobalData(EnableSidEater) & ~0x1) != 0)
     {
     LogMessage(LOG_ERR, "EnableSidEater must be 0 or 1");
     exit(0);
     }

   if ((TicketGlobalData(EnableTicket) & ~0x1) != 0)
     {
     LogMessage(LOG_ERR, "EnableTicket must be 0 or 1");
     exit(0);
     }
```

76

```
if ((TicketGlobalData(EnableLocalAuth) & ~0x1) != 0)
  {
  LogMessage(LOG_ERR, "EnablLocalAuth must be 0 or 1");
  exit(0);
  }

}
```

77

```
/*
 *--------------------------------------------------------------------
 *
 * TICKET_DebugHooks
 *
 *      Check for debug hooks and execute if found.
 *
 * Results:
 *      None.
 *
 * Side effects:
 *      None.
 *
 *--------------------------------------------------------------------
 */
static void TICKET_DebugHooks(ClientData clientData, char *suffix,
                              HTTP_Request *reqPtr)
    {
    if(strcmp(suffix, "/ticketstatus") == 0)
      {
      DumpStatus(reqPtr);
      HT_FinishRequest(reqPtr);
      return;
      }
    HTTP_Error(reqPtr, NOT_FOUND, "access denied due to poorly formed url");
    HT_FinishRequest(reqPtr);
    return;
    }
```

78

```
/*
 *----------------------------------------------------------------------
 *
 * DumpStatus --
 *
 *      Dump the server's ticket stat's
 *
 * Results:
 *      None.
 *
 * Side effects:
 *      None.
 *
 *----------------------------------------------------------------------
 */
#define BUFSIZE 1024
static void DumpStatus(HTTP_Request *reqPtr)
{
    HTTP_Server *serverPtr = reqPtr->serverPtr;
    char tmp[BUFSIZE], timeStr[BUFSIZE];
    struct utsname sysinfo;
    time_t uptime;
    int hours;


    HTTP_BeginHeader(reqPtr, "200 OK");
    HTTP_SendHeader(reqPtr, "Content-type: text/html", NULL);
    HTTP_EndHeader(reqPtr);
    HTTP_Send(reqPtr, "<title>WebServer Ticket Status</title>",
                      "<h1>WebServer Ticket Status</h1>", NULL);

    HTTP_Send(reqPtr, "<p><hr><p><h2>Ticket Log</h2>","<p><pre>\n", NULL);

    sprintf(tmp, "     <b>%s:  </b>  %d\n", "Number of access          ", Ticket
    HTTP_Send(reqPtr, tmp, NULL);
    sprintf(tmp, "     <b>%s:  </b>  %d\n", "Number of SID URL's       ", Ticket
    HTTP_Send(reqPtr, tmp, NULL);
    sprintf(tmp, "     <b>%s:  </b>  %d\n", "Number of Valid SID's     ", Ticket
    HTTP_Send(reqPtr, tmp, NULL);
    sprintf(tmp, "     <b>%s:  </b>  %d\n", "Number of Expired SID's   ", Ticket
    HTTP_Send(reqPtr, tmp, NULL);
    sprintf(tmp, "     <b>%s:  </b>  %d\n", "Number of Invalid SID's   ", Ticket
    HTTP_Send(reqPtr, tmp, NULL);
    sprintf(tmp, "     <b>%s:  </b>  %d\n", "Number of XDomain accesses", Ticket
    HTTP_Send(reqPtr, tmp, NULL);
    sprintf(tmp, "     <b>%s:  </b>  %d\n", "Number of Local Redirects ", Ticket
    HTTP_Send(reqPtr, tmp, NULL);
    sprintf(tmp, "     <b>%s:  </b>  %d\n", "Number of Remote Redirects", Ticket
    HTTP_Send(reqPtr, tmp, NULL);
    sprintf(tmp, "     <b>%s:  </b>  %d\n", "Number of No Auth servers ", Ticket


    HTTP_Send(reqPtr, tmp, "</pre>", NULL);


    uptime = time(NULL) - serverPtr->started;
    uname(&sysinfo);
    strftime(timeStr, BUFSIZE, "%A, %d-%b-%y %T",
        localtime(&serverPtr->started));
    sprintf(tmp, "Server running on <b>%s</b> (%s %s) port %d, has been up \
                 since %s.<p>", sysinfo.nodename, sysinfo.sysname,
                   sysinfo.release, serverPtr->server_port, timeStr);
    HTTP_Send(reqPtr, tmp, NULL);

    sprintf(tmp, "     <b>Number of connections:     </b> %d\n",
            serverPtr->numConnects);
    HTTP_Send(reqPtr, "<p><pre>\n", tmp, NULL);

    sprintf(tmp, "     <b>Number of HTTP requests:   </b> %d\n",
            serverPtr->numRequests);
    HTTP_Send(reqPtr, tmp, "</pre><p>", NULL);

    hours = max(uptime / 3600, 1);
    sprintf(tmp, "This server is averaging <b>%d</b> requests per hour.<p>",
            serverPtr->numRequests/hours);
    HTTP_Send(reqPtr, tmp, NULL);

    DumpRusage(reqPtr);
/*  DumpConnections(reqPtr); */

    DNS_DumpStats(reqPtr);

    HTTP_Send(reqPtr, "<p><hr><address>", DStringValue(&ht_serverSoftware),
            "</address>\n", NULL);


    reqPtr->done = TRUE;

}
#undef BUFSIZE
```

80

```
User: morris
Host: uprism.openmarket.com
Class: uprism.openmarket.com
Job: t.t
```

What is claimed is:

1. A method of processing service requests from a client to a server system through a network, said method comprising the steps of:

forwarding a service request from the client to the server system, wherein communications between the client and server system are according to hypertext transfer protocol;

returning a session identifier from the server system to the client; and

appending as part of a path name in a uniform resource locator the session identifier to the request and to subsequent service requests from the client to the server system within a session of requests.

2. A method as claimed in claim 1 wherein the session identifier includes a user identifier.

3. A method as claimed in claim 1 wherein the session identifier includes an expiration time for the session.

4. A method as claimed in claim 1 wherein the server system records information from the session identifier in a transaction log in the server system.

5. A method as claimed in claim 4 wherein the server system tracks the access history of sequences of service requests within a session of requests.

6. A method as claimed in claim 5 wherein the server system tracks the access history to determine service requests leading to a purchase made within the session of requests.

7. A method as claimed in claim 4 wherein the server system counts requests to particular services exclusive of repeated requests from a common client.

8. A method as claimed in claim 4 wherein the server system maintains a data base relating customer information to access patterns.

9. A method as claimed in claim 8 wherein the information includes customer demographics.

10. A method as claimed in claim 1 wherein the server system assigns the session identifier to an initial service request to the server system.

11. A method as claimed in claim 1 wherein the server system subjects the client to an authorization routine prior to issuing the session identifier and the session identifier is protected from forgery.

12. A method as claimed in claim 1 wherein the server system comprises plural servers including an authentication server which provides session identifiers for service requests to multiple servers.

13. A method as claimed in claim 12 wherein:

a client directs a service request to a first server which is to provide the requested service;

the first server checks the service request for a session identifier and only services a service request having a valid session identifier, and where the service request has no valid identifier:

the first server redirects the service request from the client to the authorization server;

the authorization server subjects the client to the authorization routine and issues the session identifier to be appended to the service request to the first server;

the client forwards the service request appended with the session identifier to the first server; and

the first server recognizes the session identifier and services the service request to the client; and

the client appends the session identifier to subsequent service requests to the server system and is serviced without further authorization.

14. A method as claimed in claim 13 wherein the session identifier includes a user identifier.

15. A method as claimed in claim 13 wherein the session identifier includes an expiration time for the session.

16. A method as claimed in claim 13 wherein the session identifier provides access to a protected domain to which the session has access authorization.

17. A method as claimed in claim 16 wherein the session identifier is modified for access to a different protected domain.

18. A method as claimed in claim 13 wherein the session identifier provides a key identifier for key management.

19. A method as claimed in claim 13 wherein the server system records information from the session identifier in a transaction log in the server system.

20. A method as claimed in claim 13 wherein the client modifies the path name of a current uniform resource locator using relative addressing and retains the session identifier portion of the path name unmodified for successive requests in the session.

21. A method as claimed in claim 1 wherein:

the server system subjects the client to an authorization routine prior to issuing the session identifier and the session identifier is protected from forgery, records information from the session identifier in a transaction log in the server system, tracks request paths relative to hypertext pages, and maintains a data base relating customer demographics to access patterns; and

the client modifies the path name of a current uniform resource locator using relative addressing and retains the session identifier portion of the path name unmodified for successive requests in a session.

22. A method of processing service requests from a client to a server system through a network, said method comprising the steps of:

appending as part of a path name in a uniform resource locator a session identifier to the request, wherein communications between the client and server system are according to hypertext transfer protocol;

responding to requests for hypertext pages received from a client through the network by returning the requested hypertext pages to the client;

responding to further client requests related to links in the hypertext pages; and tracking the further client requests related to a particular hypertext page.

23. A method as claimed in claim 22 wherein the requests include a common session identifier and the server system tracks client requests within a session of requests.

24. A method of processing service requests from a client to a server system through a network, said method comprising the steps of:

appending a session identifier to the request as part of a path name in a uniform resource locator, wherein communications between the client and server system are according to hypertext transfer protocol; and

responding to requests for documents received from the client through the network by returning the requested documents wherein the documents are customized for a particular user based on a user profile.

25. A method of processing service requests from a client to a server system through a network, said method comprising the steps of:

responding to a request for a document received from the client through the network, wherein communications between the client and server system are according to hypertext transfer protocol;

appending a session identifier, which includes a user identification to the request as part of a path name in a uniform resource locator; and

returning the requested document wherein the document is customized for a particular user based on the user identification of the session identifier.

26. A method of processing service requests from a client to a server system through a network, said method comprising the steps of:

appending a session identifier to the request as part of a path name in a uniform resource locator, wherein communications between the client and server system are according to hypertext transfer protocol;

responding to requests for information received from the client through the network by returning the requested information to the client; and

counting requests to particular information exclusive of repeated requests from a common client.

27. A method as claimed in claim 26 comprising excluding from the counting requests made for information from the client within a defined period of time.

28. A method of processing service requests for a document received from a client through a network in which the document has been purchased by a user, said method comprising the steps of:

responding to a request for a document received from a client through the network in which the document has been purchased by the user wherein communications between the client and server system are according to hypertext transfer protocol;

appending an authorization identifier to the request as part of a path name in a uniform resource locator; and

returning the requested document if the authorization identifier indicates that the user is authorized to access the document.

29. A method as claimed in claim 28, wherein the authorization identifier is encoded within a session identifier which is appended to the request as part of a path name in a uniform resource locator.

30. A method of processing service requests from a client to a server system through a network, said method comprising the steps of:

responding to a request for a document received from a client through the network, wherein communications between the client and server system are according to hypertext transfer protocol;

appending as part of a path name in a uniform resource locator a session identifier to the request;

returning the requested document to the client, and;

charging the user identified in the session identifier for access to the document.

31. A method as claimed in claim 30, wherein a user identifier is encoded within a session identifier which is appended to the request.

32. An information system on a network, comprising:

means for receiving service requests from clients and for determining whether a service request includes a session identifier, wherein communications between the client and server system are according to hypertext transfer protocol;

means for appending the session identifier as part of a path name in a uniform resource locator in response to an initial service request in a session of requests; and

means for servicing service requests from a client which include the session identifier, the subsequent service request being processed in the session.

33. An information system as claimed in claim 32 wherein the means for providing the session identifier is in a server system which services the requests.

34. An information system as claimed in claim 32 further comprising an authorization routine for authorizing the client prior to issuing the session identifier and means for protecting the session identifier from forgery.

35. An information server system as claimed in claim 32 further comprising a transaction log for recording information from the session identifier.

36. An information system as claimed in claim 32 further comprising means for tracking access history of sequences of service requests within the session of requests.

37. An information system as claimed in claim 32 further comprising means for counting requests to particular services exclusive of repeated requests from a common client.

38. An information system as claimed in claim 32 further comprising a data base relating customer information to access patterns.

39. An information system as claimed in claim 38 wherein the information includes customer demographics.

40. An information server on a network, comprising:

means for appending a session identifier as part of a path name in a uniform resource locator, wherein communications between the client and server system are according to hypertext transfer protocol;

means for responding to requests for hypertext pages received from a client through the network by returning the requested hypertext pages to the client;

means for responding to further requests derived from links in the hypertext pages; and means for tracking the further requests derived from a particular hypertext page.

41. A server as claimed in claim 40 wherein the requests include a common session identifier and the server tracks requests within a session of requests.

42. A server as claimed in claim 41 further comprising a data base relating customer demographics to access patterns.

43. An information server on a network, comprising:

means for appending the session identifier as part of a path name in a uniform resource locator, wherein communications between the client and server system are according to hypertext transfer protocol;

means for responding to requests for service received from a client through a network by returning the requested service to the client; and

means for counting requests to particular service exclusive of repeated requests from a common client.

44. A server as claimed in claim 43 wherein the requests include a common session identifier and the server tracks requests within a session of requests.

45. A server as claimed in claim 43 further comprising means for excluding requests made to a service from the client within a defined period of time.

* * * * *

Exhibit “B”

US005715314A

United States Patent [19]

Payne et al.

[11] Patent Number: 5,715,314
[45] Date of Patent: Feb. 3, 1998

[54] NETWORK SALES SYSTEM

[75] Inventors: Andrew C. Payne, Lincoln; Lawrence C. Stewart, Burlington; David J. Mackie, Cambridge, all of Mass.

[73] Assignee: Open Market, Inc., Cambridge, Mass.

[21] Appl. No.: 328,133

[22] Filed: Oct. 24, 1994

[51] Int. Cl.^{6} H04L 9/00
[52] U.S. Cl. 380/24; 380/23; 380/25; 380/49; 380/50
[58] Field of Search 380/4, 21, 23, 380/24, 25, 49, 50; 364/401, 406, 408, 284.4; 235/379, 380; 395/200.01, 200.02, 200.09, 925

[56] References Cited

U.S. PATENT DOCUMENTS

4,305,059	12/1981	Benton	340/825.33
4,578,530	3/1986	Zeidler .	
4,734,858	3/1988	Schlafly	364/408
4,755,940	7/1988	Brachtl et al.	364/408
4,775,935	10/1988	Yourick	364/401
4,795,890	1/1989	Goldman	235/380
4,799,156	1/1989	Shavit et al.	364/401
4,812,628	3/1989	Boston et al.	235/380
4,827,508	5/1989	Shear	380/4
4,922,521	5/1990	Krikke et al.	379/95
4,935,870	6/1990	Burk, Jr. et al. .	
4,947,028	8/1990	Gorog	235/381
4,977,595	12/1990	Ohta et al.	380/24
4,982,346	1/1991	Girouard et al.	364/550
4,992,940	2/1991	Dworkin	364/401
5,025,373	6/1991	Keyser, Jr. et al.	364/408
5,060,153	10/1991	Nakagawa	364/405
5,077,607	12/1991	Johnson et al. .	

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

0-542-298-A2	5/1993	European Pat. Off.	G07F 7/10
2102606	2/1983	United Kingdom	G07F 7/10
WO 91/16691	10/1991	WIPO	G07F 7/10
WO 95/16971	6/1995	WIPO .	

OTHER PUBLICATIONS

Rivest, R.L. et al., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, no date.
Bellcore Internal E-Mail, Nov. 24, 1993.
Sirbu, Marvin A.; "Internet Billing Service Design and Prototype Implementation"; *An Internet Billing Server*; pp. 1-19, no date.
Payment Systems, "United States"; pp. 115-135, no date.
National Westminster Bank Group Brochure; pp. 1-29, no date.

(List continued on next page.)

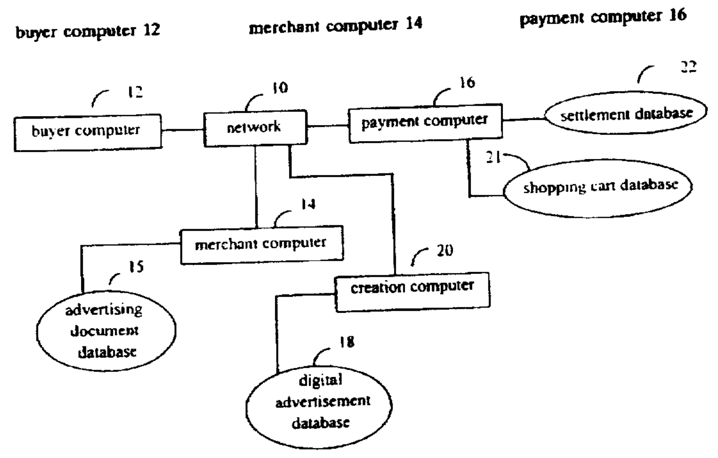
Primary Examiner—Bernarr E. Gregory
Attorney, Agent, or Firm—Fish & Richardson P.C.

[57] ABSTRACT

A network-based sales system includes at least one buyer computer for operation by a user desiring to buy a product, at least one merchant computer, and at least one payment computer. The buyer computer, the merchant computer, and the payment computer are interconnected by a computer network. The buyer computer is programmed to receive a user request for purchasing a product, and to cause a payment message to be sent to the payment computer that comprises a product identifier identifying the product. The payment computer is programmed to receive the payment message, to cause an access message to be created that comprises the product identifier and an access message authenticator based on a cryptographic key, and to cause the access message to be sent to the merchant computer. The merchant computer is programmed to receive the access message, to verify the access message authenticator to ensure that the access message authenticator was created using the cryptographic key, and to cause the product to be sent to the user desiring to buy the product.

48 Claims, 25 Drawing Sheets

Microfiche Appendix Included
(1 Microfiche, 34 Pages)



U.S. PATENT DOCUMENTS

5,220,501	6/1993	Lawlor et al.	364/408
5,247,575	9/1993	Sprague et al.	380/9
5,305,195	4/1994	Murphy	364/401
5,336,870	8/1994	Hughes	235/379
5,341,429	8/1994	Stringer et al.	380/23
5,347,632	9/1994	Filepp et al.	395/200.09
5,351,186	9/1994	Bullock et al.	364/401
5,351,293	9/1994	Michener et al.	380/21
5,383,113	1/1995	Kight et al.	364/401
5,414,833	5/1995	Hershey et al.	395/575

OTHER PUBLICATIONS

Even et al.; "Electronic Wallet"; pp. 383–386; 1983.

Okamoto et al.; "Universal Electronic Cash"; pp. 324–337; 1991.

Pfitzmann et al.; "How to Break and Repair a 'Provably Secure' Untraceable Payment System"; pp. 338–350; 1991.

Intuit Corp Quicken User's Guide; "Paying Bills Electronically"; pp. 171–192, no date.

Compuserve International; Compuserve Information Service Users Guide; pp. 109–114; 1986.

Gifford, David; "Notes on Community Information Systems" MIT LCS TM–419; Dec., 1989.

Vittal, J. "Active Message Processing: Messages as Messengers"; pp. 175–195; 1981.

Bos et al.; "SmartCash: A Practical Electronic Payment System"; pp. 1–8; Aug. 1990.

American National Standard; "Financial Institution Retail Message Authentication"; ANSI X9.19; 1986.

American National Standard; "Interchange Message Specification for Debit and Credit Card Message Exchange Among Financial Institutions"; ANSI X9.2; 1988.

Chaum et al., "Achieving Electronic Privacy"; *Scientific American*; pp. 319–327; 1988.

Bürk et al.; "Value Exchange Systems Enabling Security and Unobservability"; *Computers & Security*, 9; pp. 715–721; 1990.

Chaum et al.; "Untraceable Electronic Cash"; *Advances in Cryptology*; pp. 319–327; 1988.

Schamüller-Bichl. I; "IC–Cards in High–Security Applications"; Selected Papers from the Smart Card 2000 Conference; *Springer Verlag*; pp. 177–199; 1991.

Newman, B.C.; "Proxy–Based Authorization and Accounting for Distributed Systems"; *Proc. 13th Int. Conf. on Dist. Comp. Sys.*; May, 1993.

Medvinsky et al.; "Electronic Currency for the Internet"; *Electronic Markets*; pp. 30–31, Sep., 1993.

Anderson, Ross J.; "UEPS—A Second Generation Electronic Wallet"; Proc. of the Second European Symposium on Research in Computer Security (ESORICS); Touluse, France; pp. 411–418, no date.

Anderson, Ross; "Why Cryptosystems Fail"; Proc. 1st Conf. Computer and Comm. Security; pp. 215–227; Nov., 1993.

Dukach, Semyon; "SNPP: A Simple Network Payment Protocol"; MIT Laboratory for Computer Science; Cambridge, Massachusetts; 1993.

Medvinsky et al.; "NetCash: A Design for Practical Electronic Currency on the Internet"; Proc. 1st ACM Conf. on Comp. and Comm. Security; Nov., 1993.

Society for Worldwide Interbank Financial Telecommunications S.C.; "A S.W.I.F.T. Overview", no date.

Case Study: The CIRRUS Banking Network; Comm. ACM 8, 28' pp. 797–8078; Aug., 1985.

Intel Corporation; Power Technology; Marketig Brochure, no date.

Bender, M.; "EFTS: Electronic Funds Transfer Systems"; Kennikat Press; Port Washington, New York; pp. 43–46; 1975.

Abadi, M. et al.; "Authentication and Delegation with Smart–Cards" Report 67; Systems Research Center; Digital Equipment Corporation; Palo Alto, California; Oct. 22, 1990, revised Jul. 30, 1992.

Information Network Institute, Carnegie Mellon University; Internet Billing Server; Prototype Scope Document; Oct. 14, 1993.

Krajewski, M.; "Concept for a Smart Card Kerberos"; 15th National Computer Security Conference; Oct., 1992.

Krajewski, M.; "Smar Card Augmentation of Kerberos"; Privacy and Security Research Group Workshop on Network and Distributed System Security; Feb., 1993.

Krajewski, M. et al.; "Applicability of Smart Cards to Network User Authentication"; *Computing Systems*; vol. 7, No. 1; 1994.

Harty et al.; "Case Study: The VISA Transaction Processing System"; 1988.

International Organization for Standardization; "International Standard: Bank Card Originated Messages—Interchange Message Specifications—Content for Financial Transactions"; ISO 8583; 1987.

Rivest, R.; "The MD5 Message–Digest Algorithm"; MIT Laboratory for Computer Science and RSA Data Security, Inc.; Apr., 1992.

Voydock, Victor et al.; "Security Mechanisms in High–Level Network Protocols"; Computer Surveys; vol. 15, No. 2; Jun., 1981.

Needham, Roger M., "Adding Capability Access to Conventional File Servers"; Xerox Palo Alto Research Center; Palo Alto, California; no date.

Gligor, Virgil D. et al.; "Object Migration and Authentication"; IEEE Transactions on Software Engineering; vol. SE–5, No. 6; Nov., 1979.

Chaum, D.L. et al.; "Implementing Capability–Based Protection Using Encryption"; Electronics Research Laboratory, College of Engineering, University of California, Berkeley, California; Jul. 17, 1978.

Gifford, David K.; "Cryptographic Sealing for Information Secrecy and Authentication"; Stanford University and Xerox Palo Alto Research Center; Communications of the ACM; vol. 25, No. 4; Apr., 1982.

Mosaic Communications Corp. press release; "Mosaic Communications Unveils Network Navigator and Server Software for the Internet"; Sep. 12, 1994.

Rescorla, E. and Schiffman, A.; "The Secure HyperText Transfer Protocol"; Enterprise Integration Technologies; Jun., 1994.

Tenenbaum, Jay M. and Schiffman, Allan M.; "Development of Network Infrastructure and Services for Rapid Acquisition"; adapted from a white paper submitted to DARPA by MCC in collaboration with EIT and ISI.

Cohen, Danny; "Computerized Commerce"; ISI Reprint Series ISI/RS–89–243; Oct., 1989; Reprinted from Information Processing 89, Proceedings of the IFIP World Computer Congress, held Aug. 28–Sep. 1 1989.

Cohen, Danny; "Electronic Commerce"; University of Southern California Information Sciences Institute, Research Report ISI/RR–89–244; Oct., 1989.

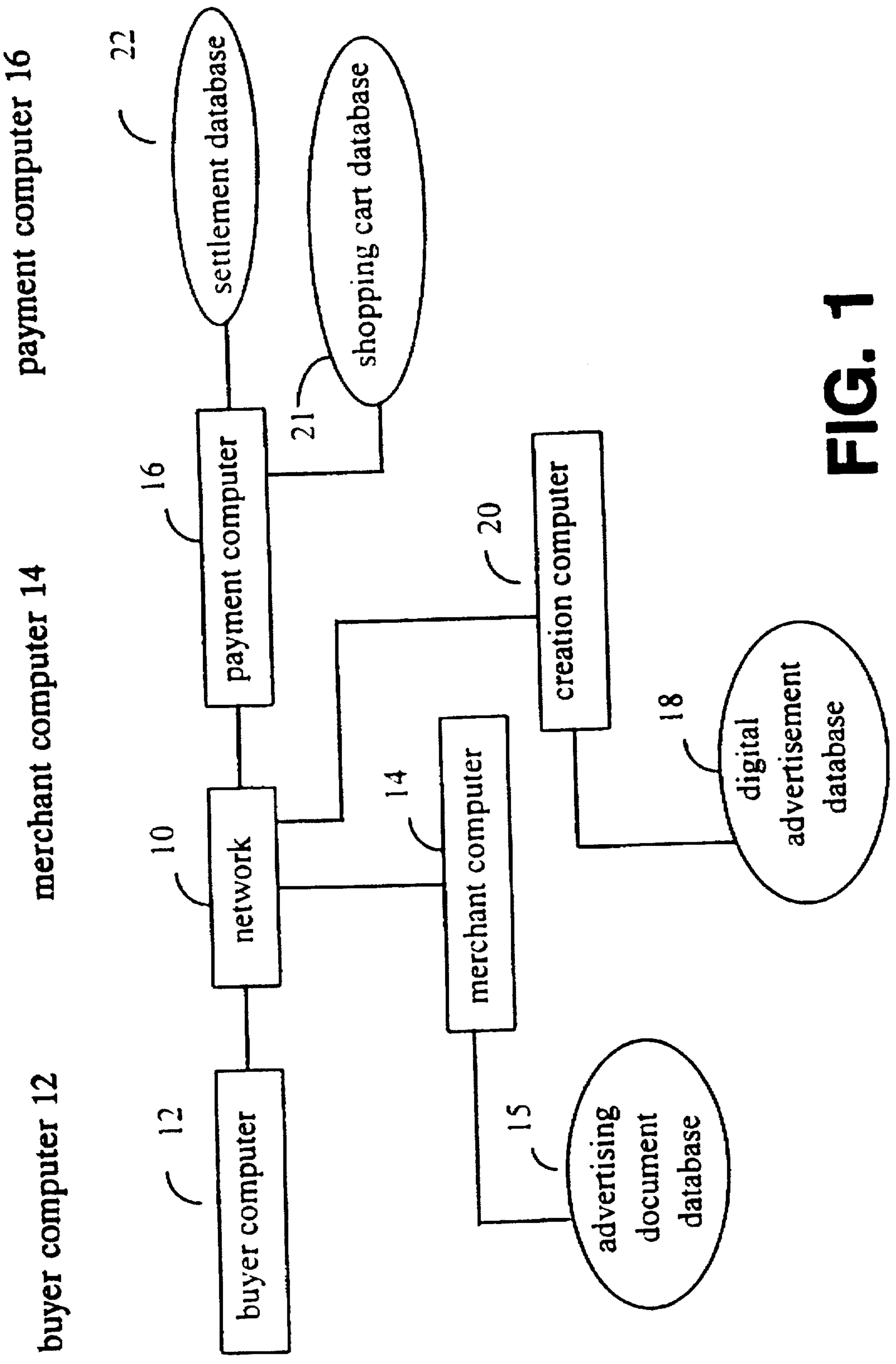

buyer computer 12
merchant computer 14
payment computer 16
buyer computer
12
network
10
payment computer
16
settlement database
22
shopping cart database
21
merchant computer
14
creation computer
20
advertising document database
15
digital advertisement database
18

FIG. 1

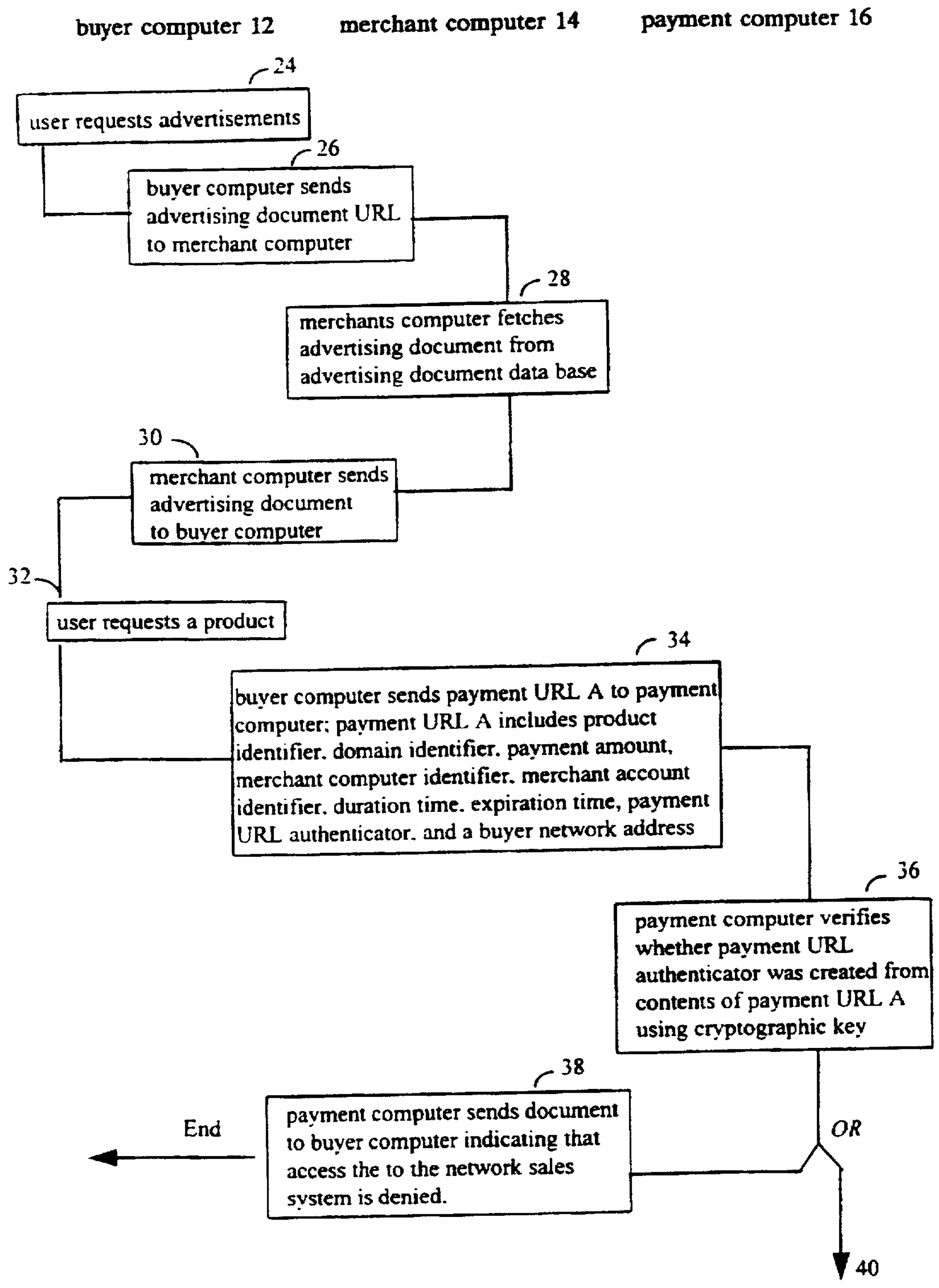

buyer computer 12
merchant computer 14
payment computer 16
24
user requests advertisements
26
buyer computer sends advertising document URL to merchant computer
28
merchants computer fetches advertising document from advertising document data base
30
merchant computer sends advertising document to buyer computer
32
user requests a product
34
buyer computer sends payment URL A to payment computer; payment URL A includes product identifier. domain identifier. payment amount, merchant computer identifier. merchant account identifier. duration time. expiration time, payment URL authenticator. and a buyer network address
36
payment computer verifies whether payment URL authenticator was created from contents of payment URL A using cryptographic key
38
payment computer sends document to buyer computer indicating that access the to the network sales system is denied.
End
OR
40

FIG. 2A

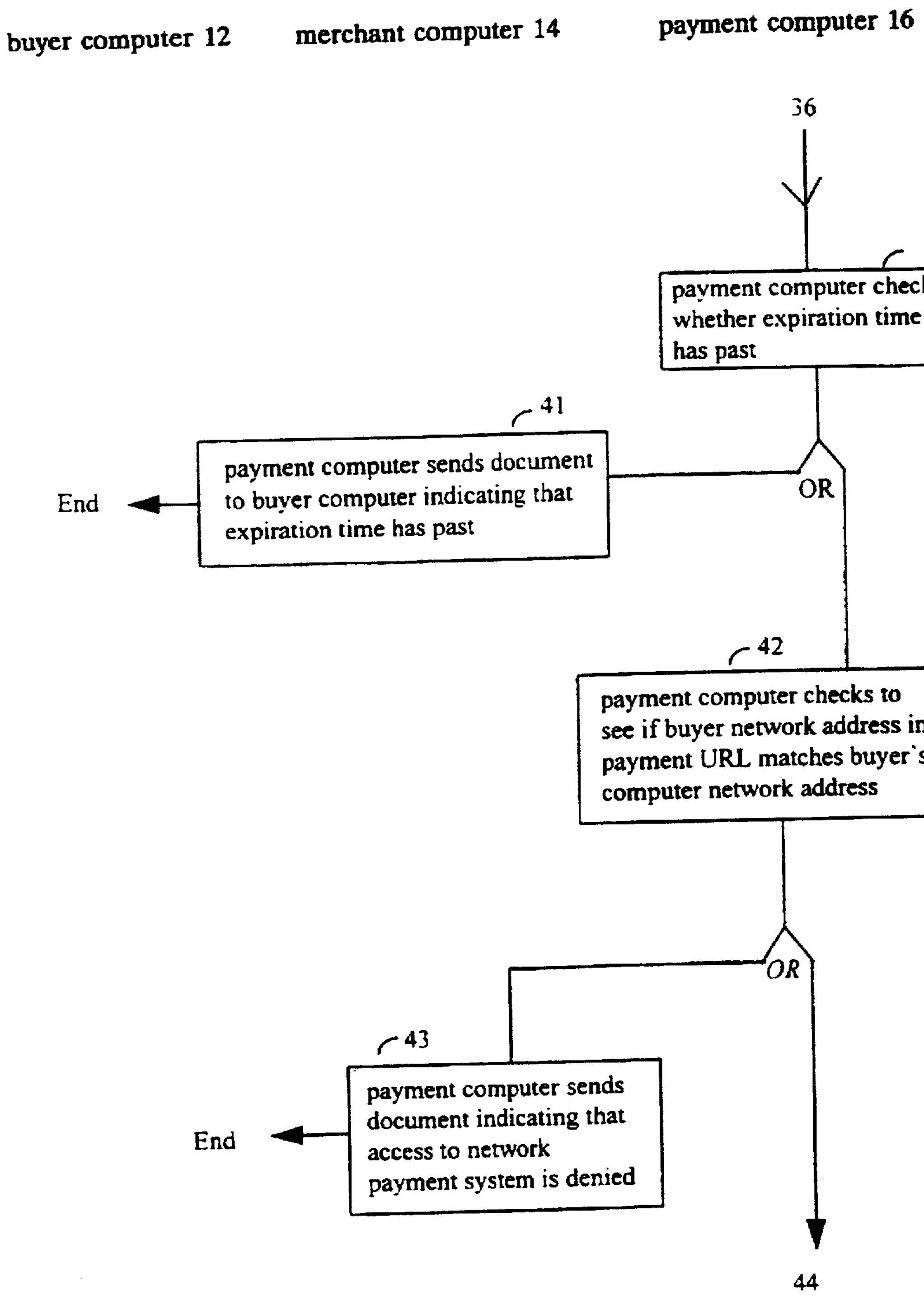

buyer computer 12
merchant computer 14
payment computer 16
36
40
payment computer checks whether expiration time has past
41
payment computer sends document to buyer computer indicating that expiration time has past
End
OR
42
payment computer checks to see if buyer network address in payment URL matches buyer's computer network address
OR
43
payment computer sends document indicating that access to network payment system is denied
End
44

FIG. 2B

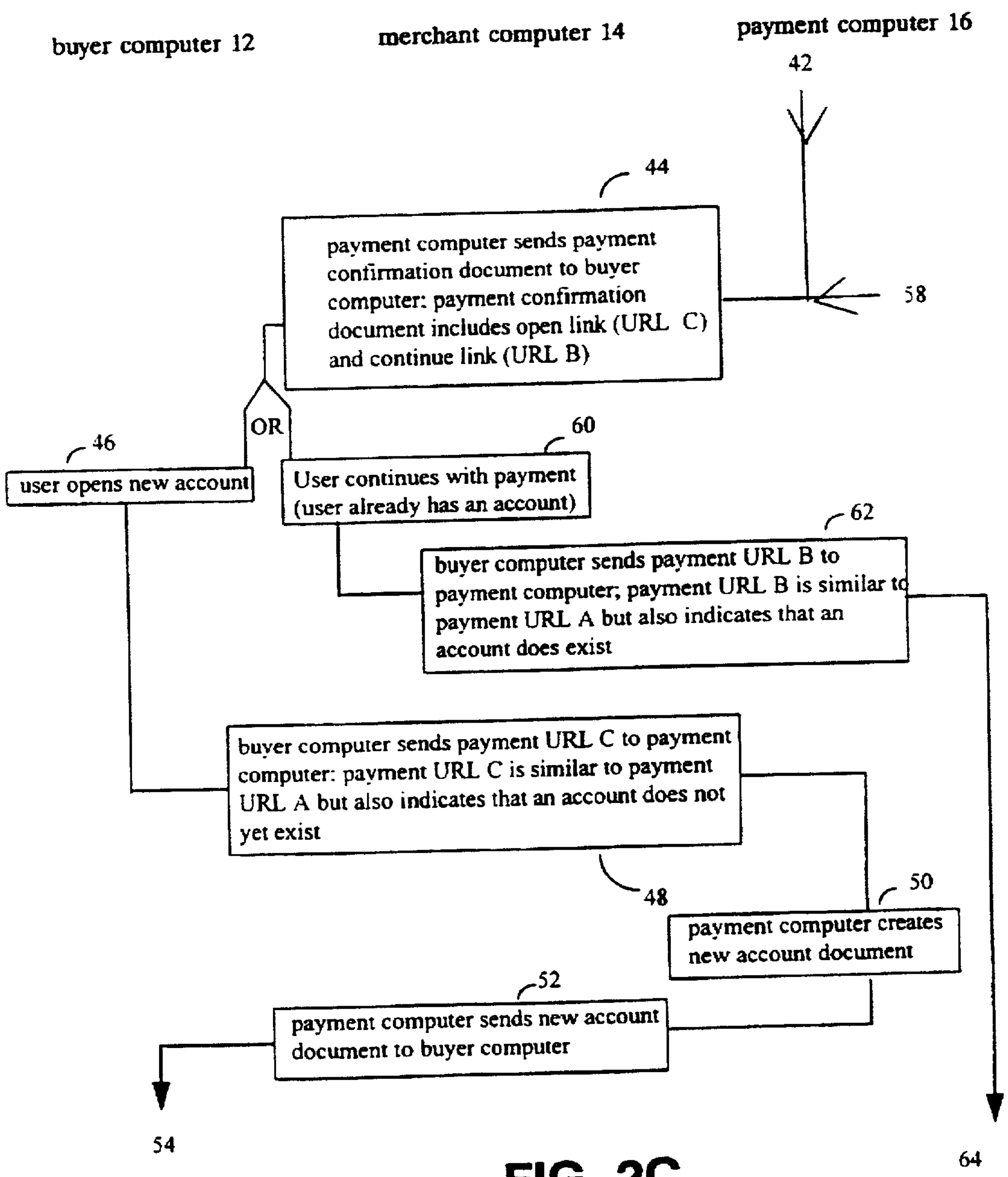

buyer computer 12
merchant computer 14
payment computer 16
42
44
payment computer sends payment confirmation document to buyer computer: payment confirmation document includes open link (URL C) and continue link (URL B)
58
OR
46
user opens new account
60
User continues with payment (user already has an account)
62
buyer computer sends payment URL B to payment computer; payment URL B is similar to payment URL A but also indicates that an account does exist
buyer computer sends payment URL C to payment computer: payment URL C is similar to payment URL A but also indicates that an account does not yet exist
48
50
payment computer creates new account document
52
payment computer sends new account document to buyer computer
54
64

FIG. 2C

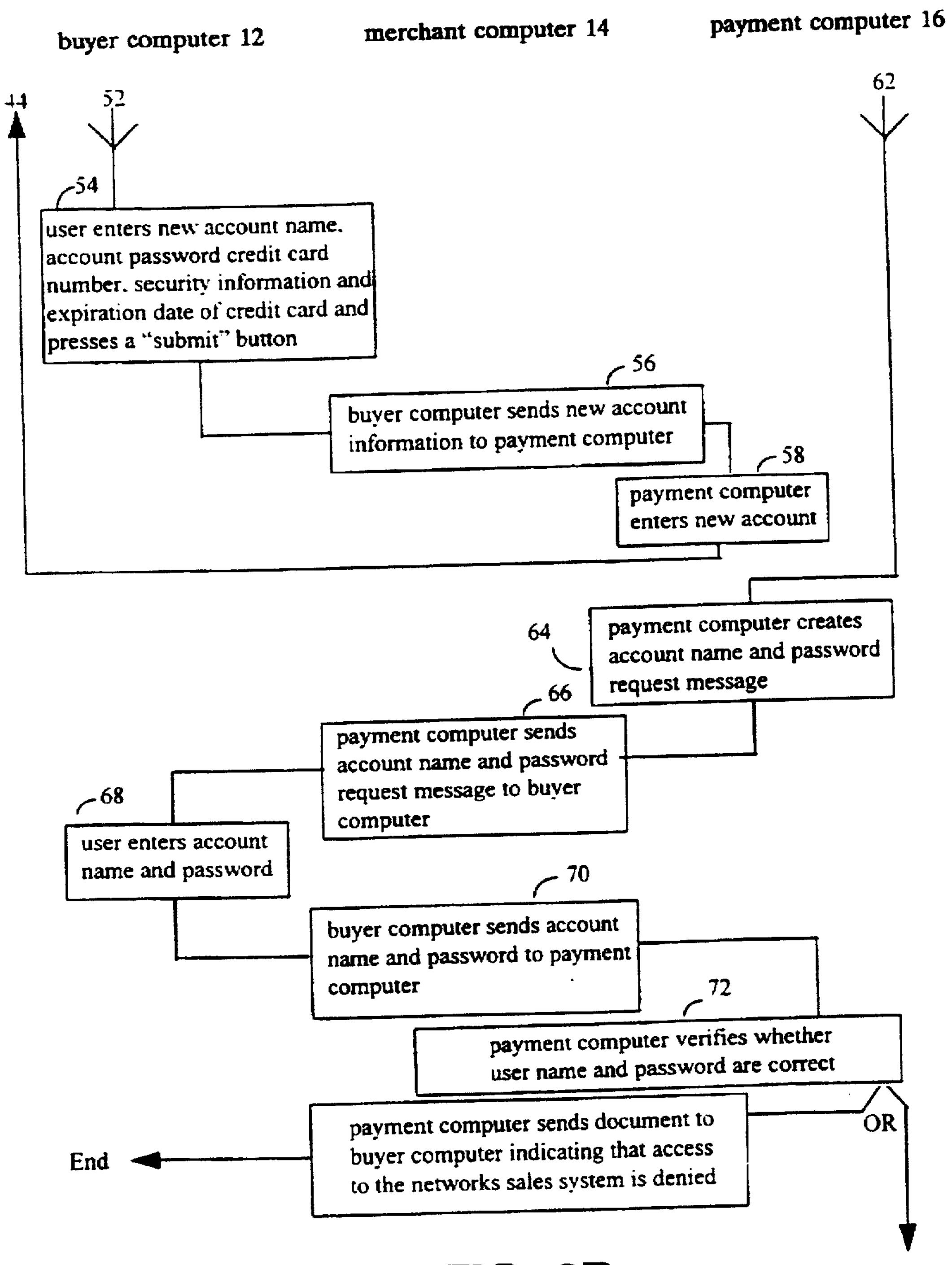

buyer computer 12
merchant computer 14
payment computer 16
44
52
62
54
user enters new account name. account password credit card number. security information and expiration date of credit card and presses a "submit" button
56
buyer computer sends new account information to payment computer
58
payment computer enters new account
64
payment computer creates account name and password request message
66
payment computer sends account name and password request message to buyer computer
68
user enters account name and password
70
buyer computer sends account name and password to payment computer
72
payment computer verifies whether user name and password are correct
OR
payment computer sends document to buyer computer indicating that access to the networks sales system is denied
End
73

FIG. 2D

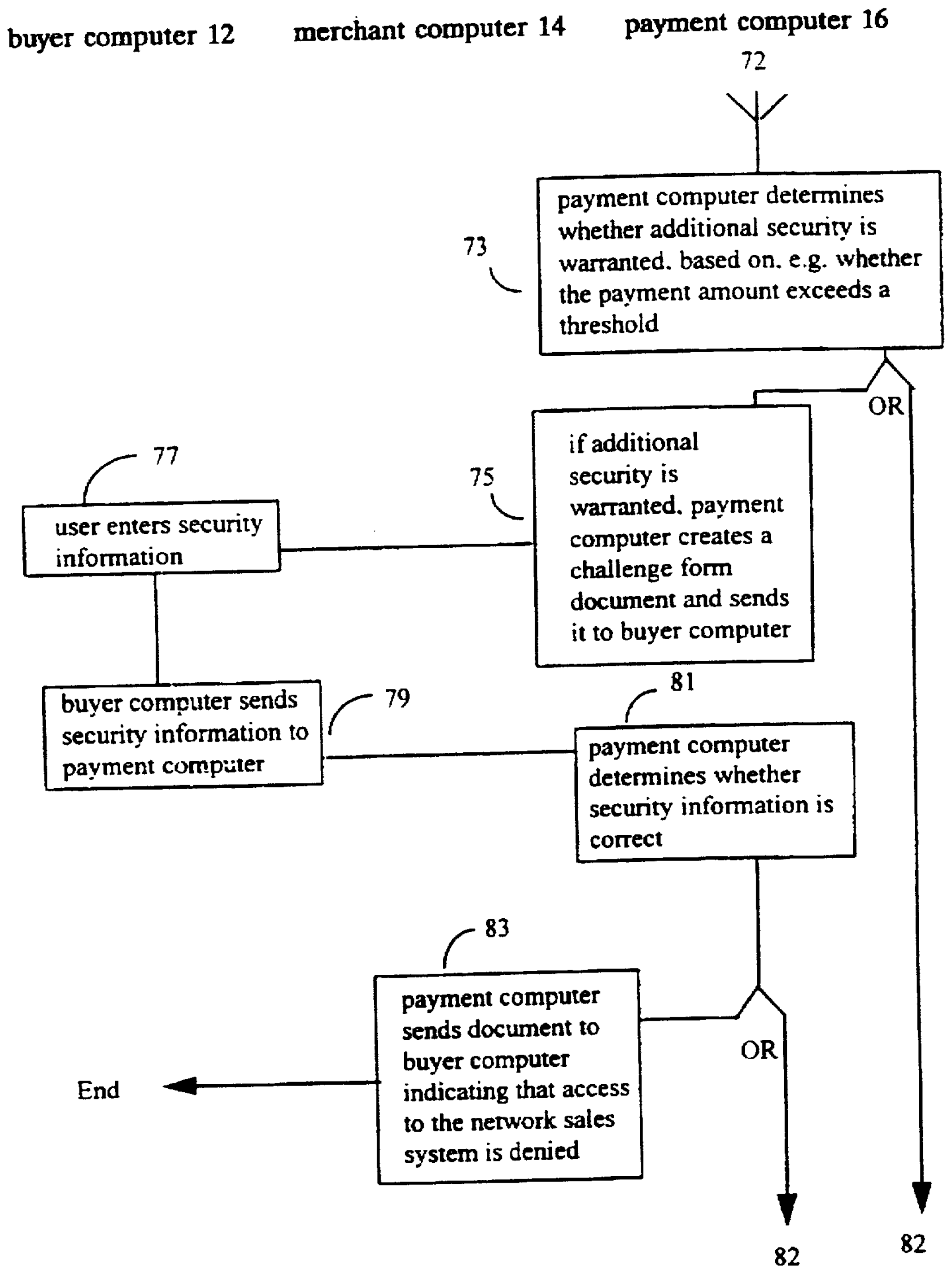

buyer computer 12
merchant computer 14
payment computer 16
72
73
payment computer determines whether additional security is warranted. based on. e.g. whether the payment amount exceeds a threshold
OR
77
user enters security information
75
if additional security is warranted. payment computer creates a challenge form document and sends it to buyer computer
buyer computer sends security information to payment computer
79
81
payment computer determines whether security information is correct
83
payment computer sends document to buyer computer indicating that access to the network sales system is denied
End
OR
82
82

FIG. 2E

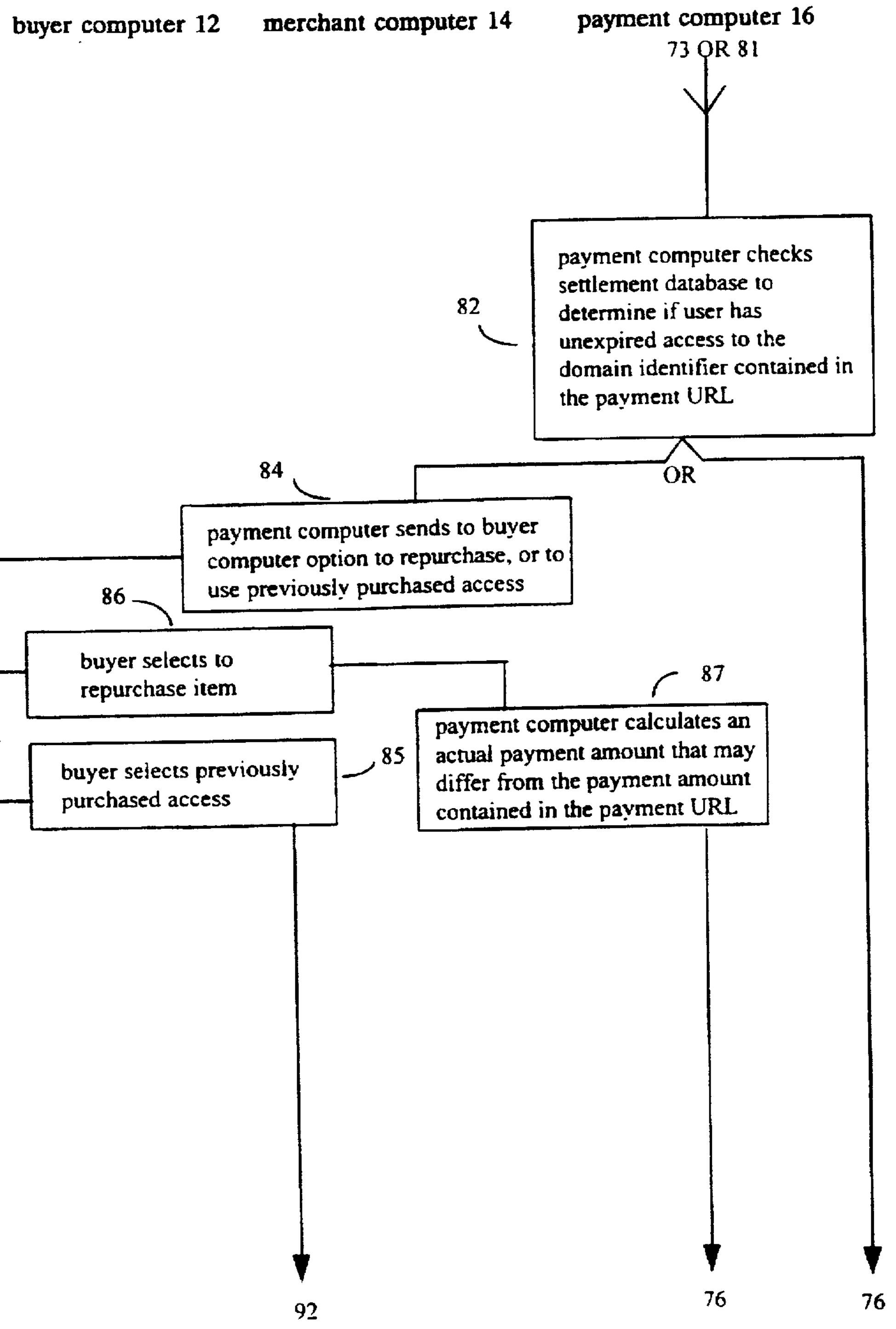

buyer computer 12
merchant computer 14
payment computer 16
73 OR 81
82
payment computer checks settlement database to determine if user has unexpired access to the domain identifier contained in the payment URL
OR
84
payment computer sends to buyer computer option to repurchase, or to use previously purchased access
86
buyer selects to repurchase item
OR
buyer selects previously purchased access
85
87
payment computer calculates an actual payment amount that may differ from the payment amount contained in the payment URL
92
76
76

FIG. 2F

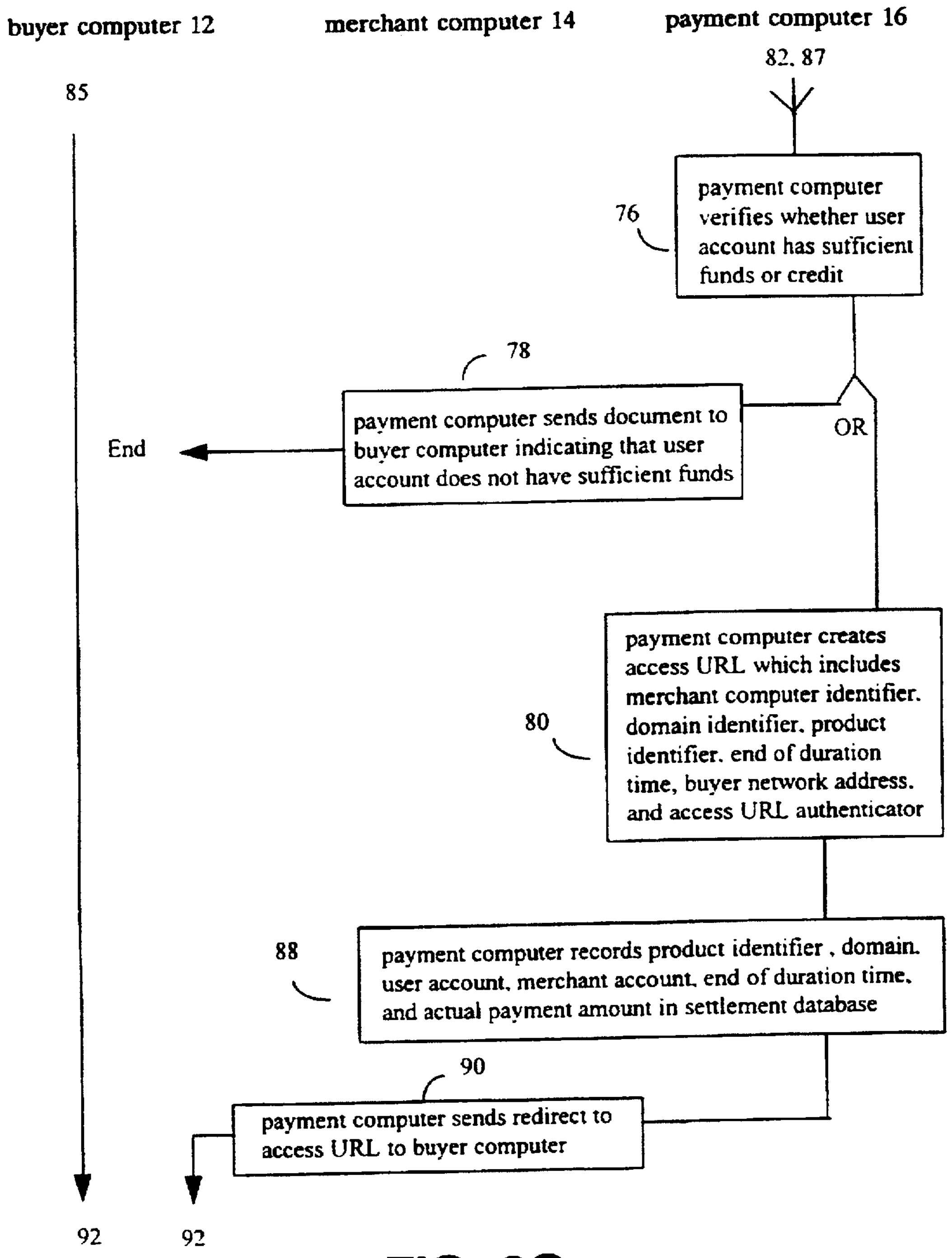

buyer computer 12
merchant computer 14
payment computer 16
82, 87
85
76
payment computer verifies whether user account has sufficient funds or credit
78
payment computer sends document to buyer computer indicating that user account does not have sufficient funds
End
OR
80
payment computer creates access URL which includes merchant computer identifier, domain identifier, product identifier, end of duration time, buyer network address, and access URL authenticator
88
payment computer records product identifier, domain, user account, merchant account, end of duration time, and actual payment amount in settlement database
90
payment computer sends redirect to access URL to buyer computer
92
92

FIG. 2G

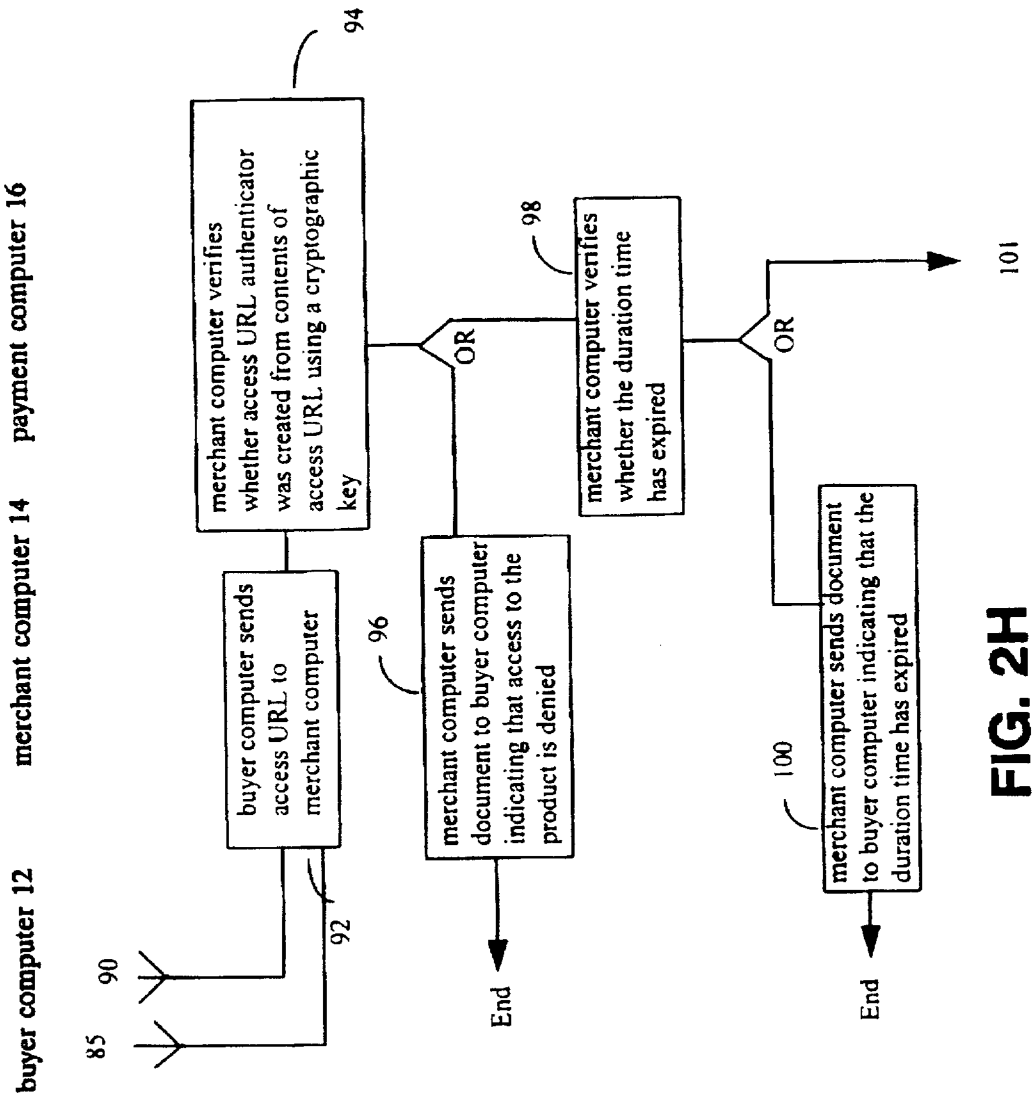

buyer computer 12
merchant computer 14
payment computer 16
85
90
buyer computer sends access URL to merchant computer
92
merchant computer verifies whether access URL authenticator was created from contents of access URL using a cryptographic key
94
OR
merchant computer sends document to buyer computer indicating that access to the product is denied
96
End
merchant computer verifies whether the duration time has expired
98
OR
101
merchant computer sends document to buyer computer indicating that the duration time has expired
100
End

FIG. 2H

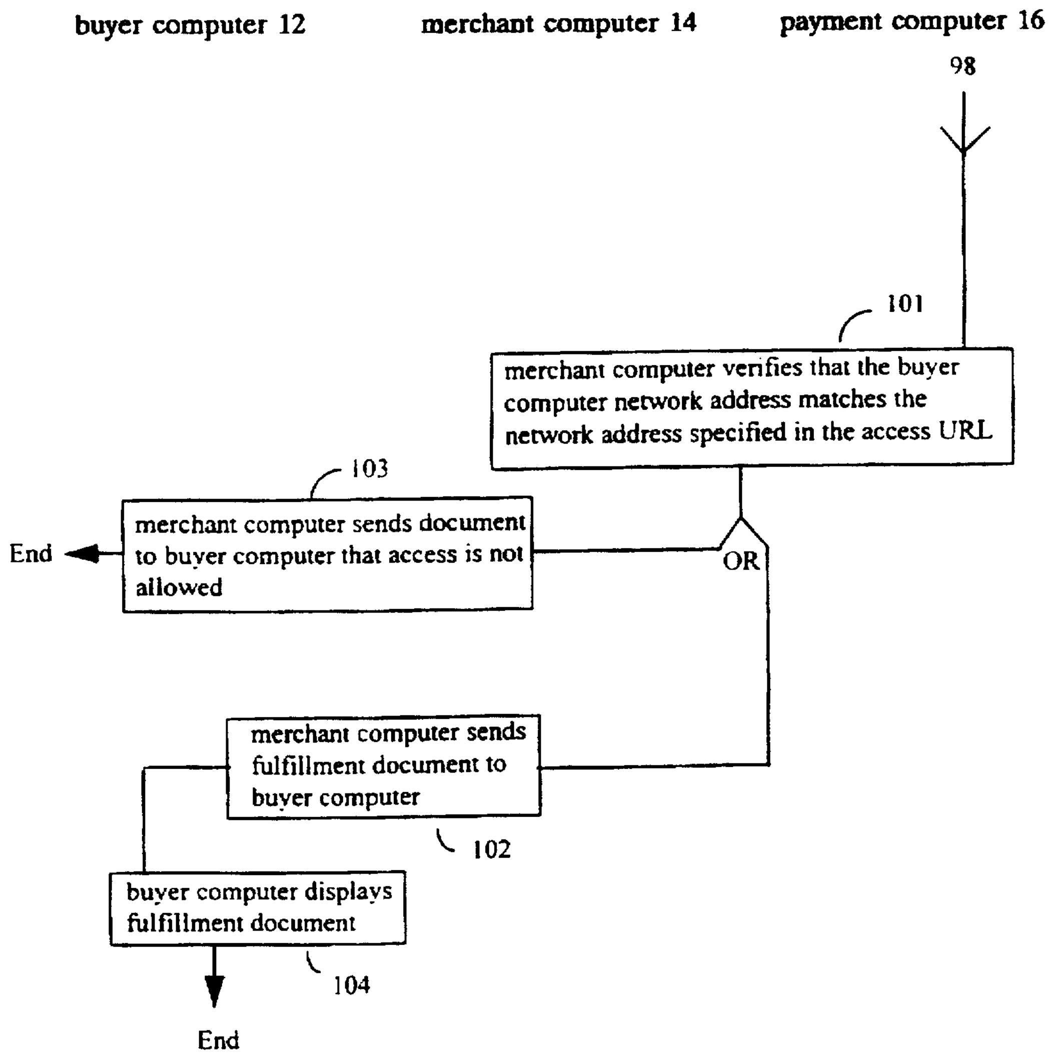

buyer computer 12
merchant computer 14
payment computer 16
98
101
merchant computer verifies that the buyer computer network address matches the network address specified in the access URL
103
merchant computer sends document to buyer computer that access is not allowed
End
OR
merchant computer sends fulfillment document to buyer computer
102
buyer computer displays fulfillment document
104
End

FIG. 2I

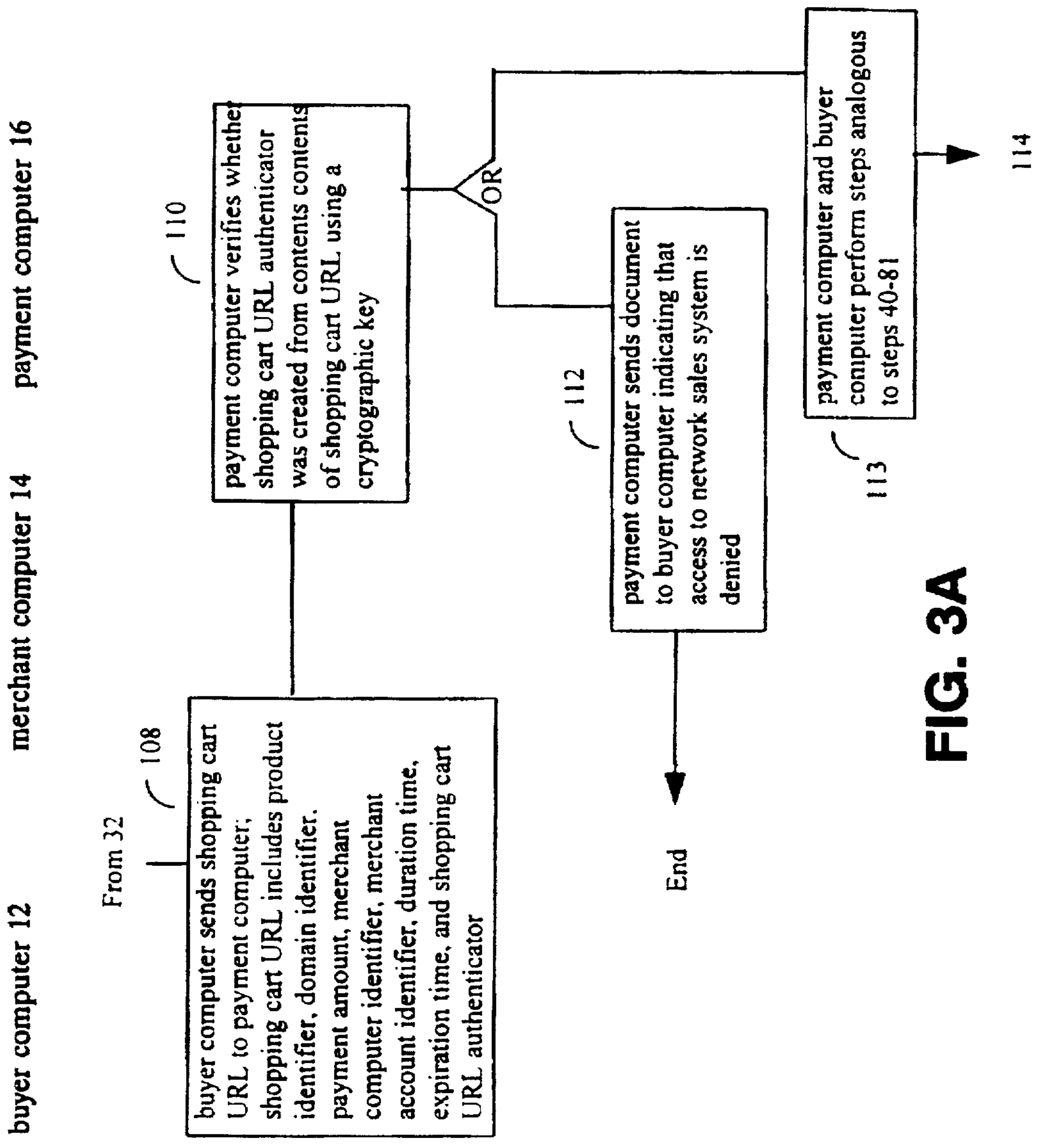
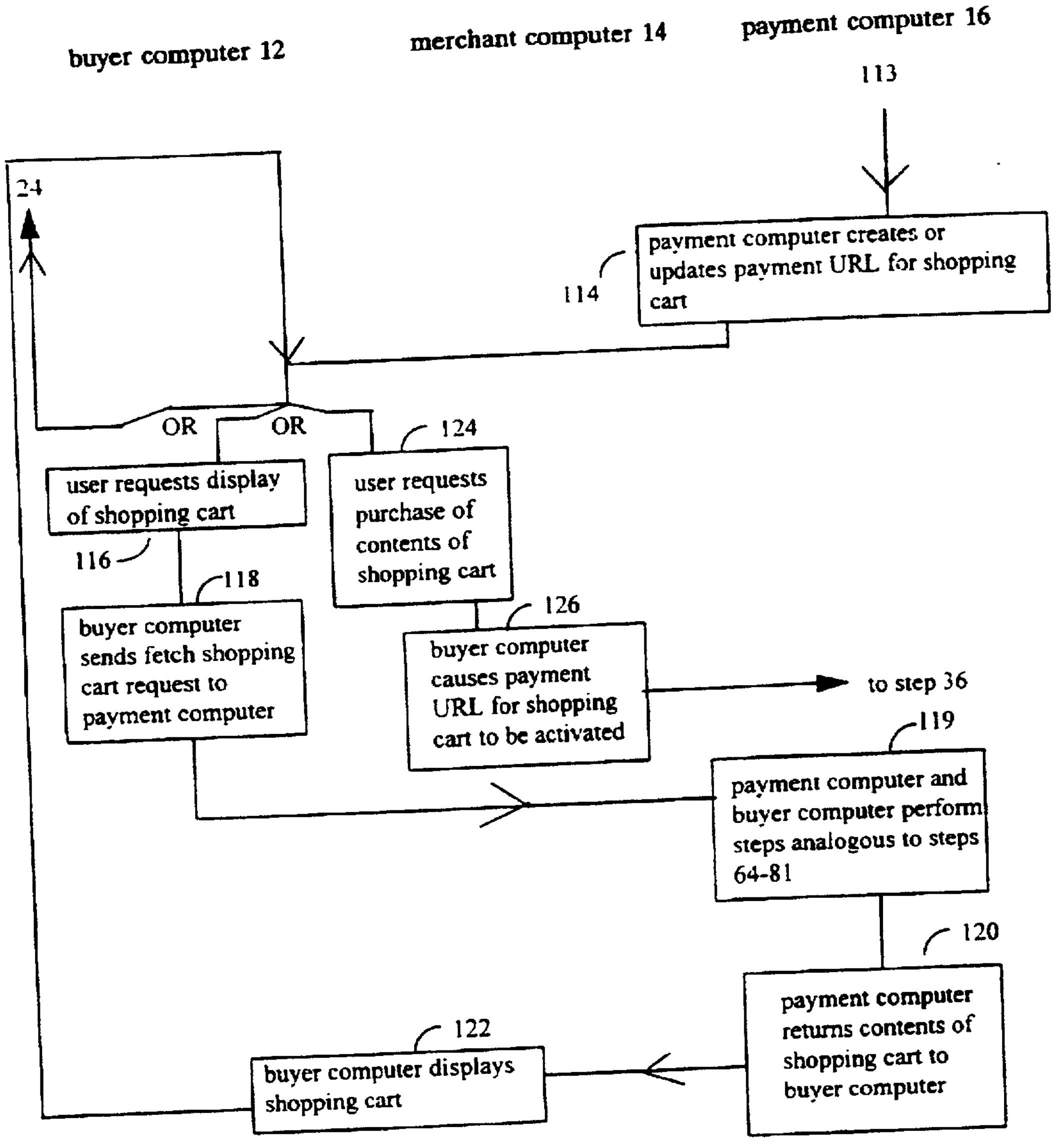

buyer computer 12
merchant computer 14
payment computer 16
From 32
108
buyer computer sends shopping cart URL to payment computer; shopping cart URL includes product identifier, domain identifier, payment amount, merchant computer identifier, merchant account identifier, duration time, expiration time, and shopping cart URL authenticator
110
payment computer verifies whether shopping cart URL authenticator was created from contents contents of shopping cart URL using a cryptographic key
OR
112
payment computer sends document to buyer computer indicating that access to network sales system is denied
End
113
payment computer and buyer computer perform steps analogous to steps 40-81
114

FIG. 3A



buyer computer 12
merchant computer 14
payment computer 16
113
24
payment computer creates or updates payment URL for shopping cart
114
OR
OR
124
user requests display of shopping cart
116
user requests purchase of contents of shopping cart
118
buyer computer sends fetch shopping cart request to payment computer
126
buyer computer causes payment URL for shopping cart to be activated
to step 36
119
payment computer and buyer computer perform steps analogous to steps 64-81
120
payment computer returns contents of shopping cart to buyer computer
122
buyer computer displays shopping cart

FIG. 3B

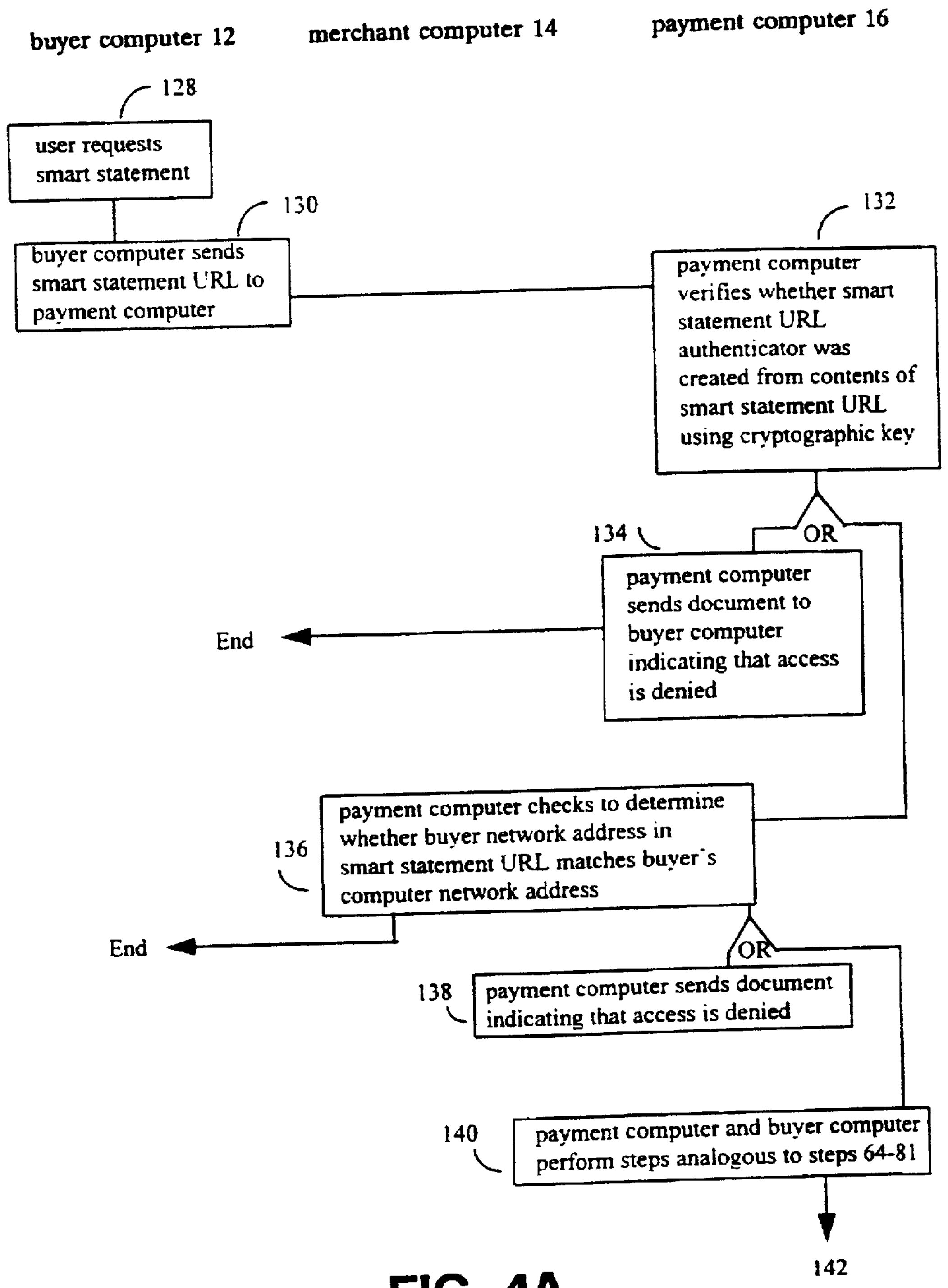

buyer computer 12
merchant computer 14
payment computer 16
128
user requests smart statement
130
buyer computer sends smart statement URL to payment computer
132
payment computer verifies whether smart statement URL authenticator was created from contents of smart statement URL using cryptographic key
OR
134
payment computer sends document to buyer computer indicating that access is denied
End
136
payment computer checks to determine whether buyer network address in smart statement URL matches buyer's computer network address
End
OR
138
payment computer sends document indicating that access is denied
140
payment computer and buyer computer perform steps analogous to steps 64-81
142

FIG. 4A

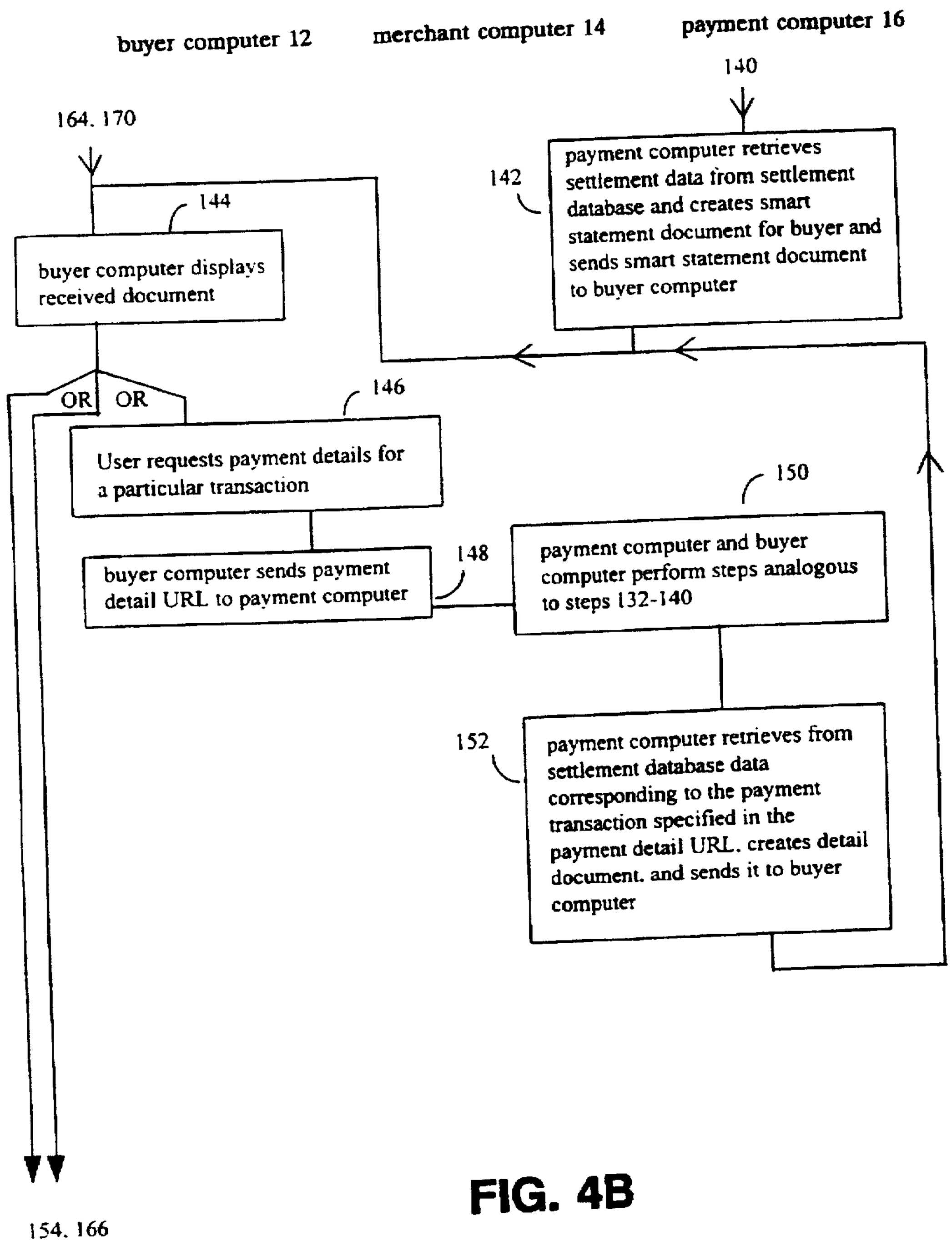

buyer computer 12
merchant computer 14
payment computer 16
140
164. 170
142
payment computer retrieves settlement data from settlement database and creates smart statement document for buyer and sends smart statement document to buyer computer
144
buyer computer displays received document
OR
OR
146
User requests payment details for a particular transaction
150
148
buyer computer sends payment detail URL to payment computer
payment computer and buyer computer perform steps analogous to steps 132-140
152
payment computer retrieves from settlement database data corresponding to the payment transaction specified in the payment detail URL. creates detail document. and sends it to buyer computer
154. 166

FIG. 4B

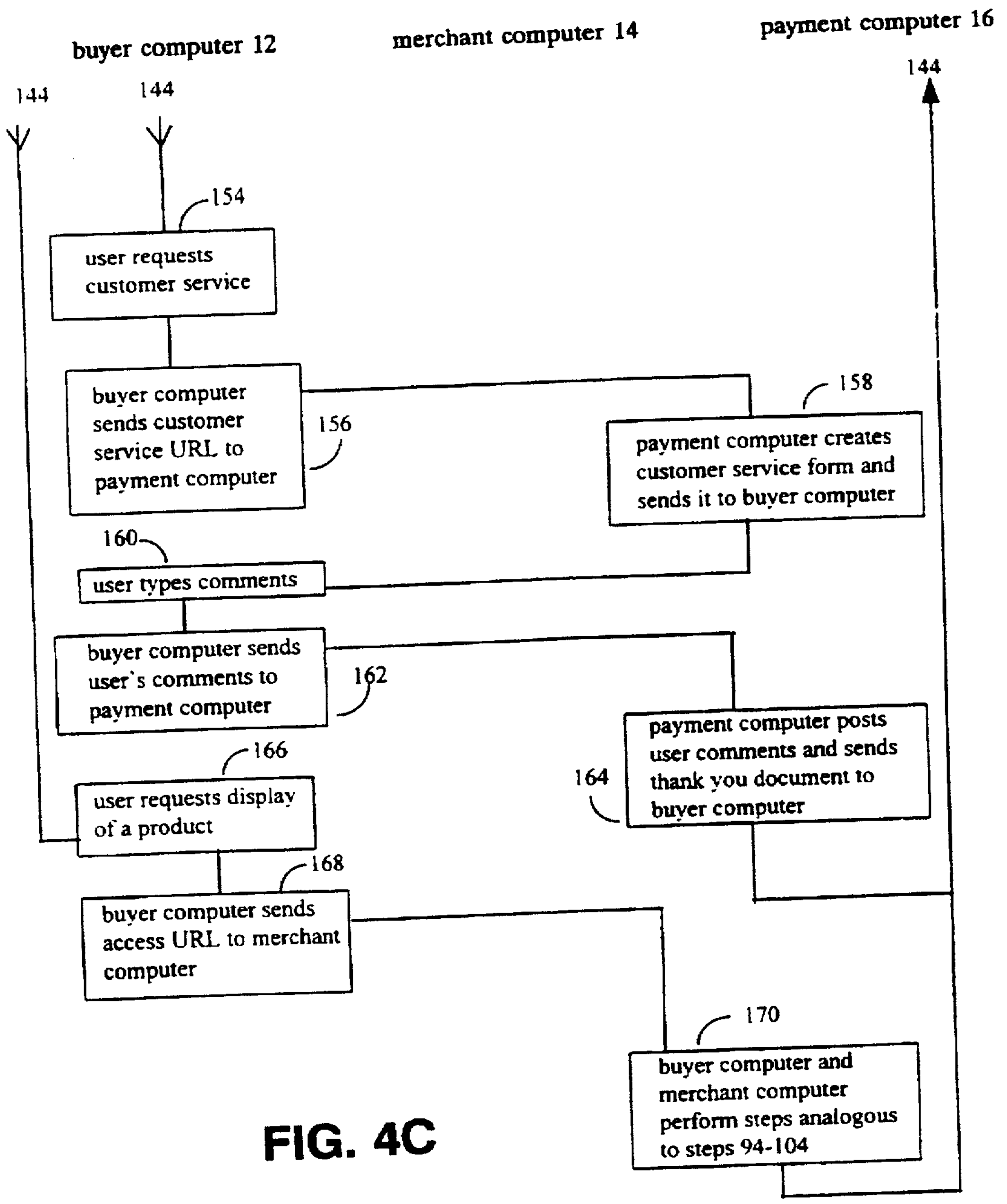
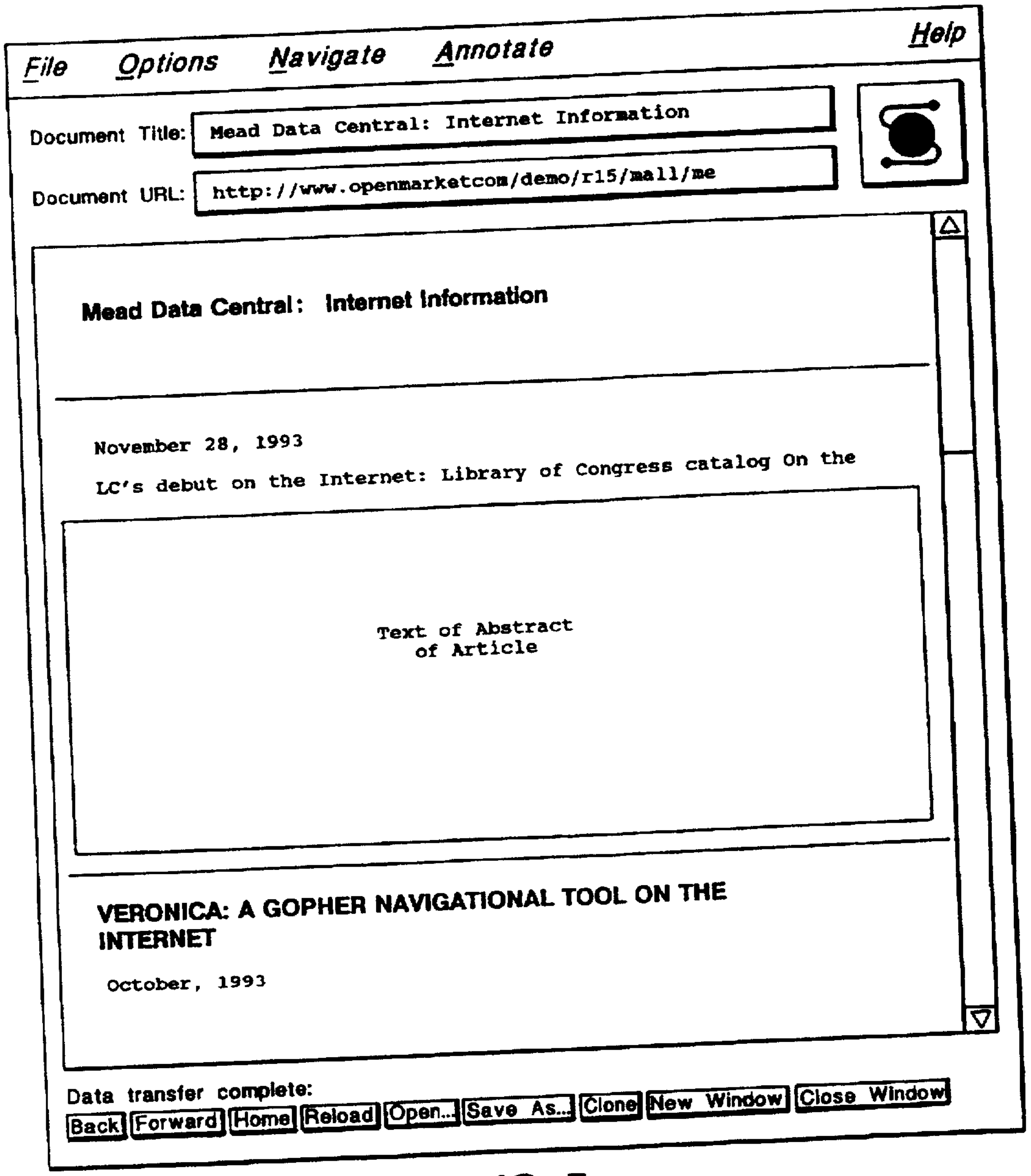

buyer computer 12
merchant computer 14
payment computer 16
144
144
144
154
user requests customer service
buyer computer sends customer service URL to payment computer
156
158
payment computer creates customer service form and sends it to buyer computer
160
user types comments
buyer computer sends user's comments to payment computer
162
payment computer posts user comments and sends thank you document to buyer computer
164
166
user requests display of a product
168
buyer computer sends access URL to merchant computer
170
buyer computer and merchant computer perform steps analogous to steps 94-104

FIG. 4C



Document Title: Mead Data Central: Internet Information

Document URL: http://www.openmarketcom/demo/r15/mall/me

Mead Data Central: Internet Information

November 28, 1993

LC's debut on the Internet: Library of Congress catalog On the

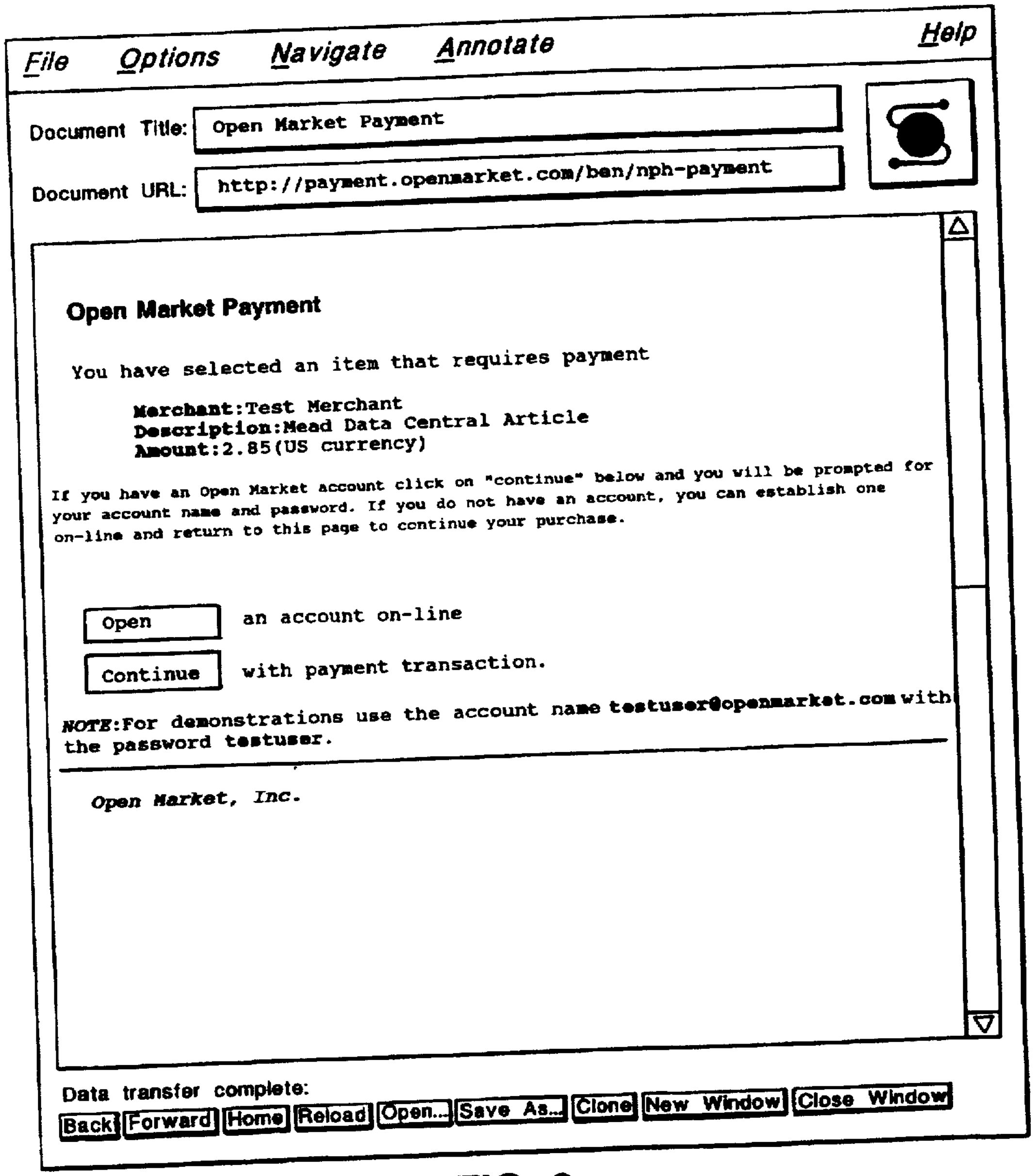
Text of Abstract
of Article

VERONICA: A GOPHER NAVIGATIONAL TOOL ON THE INTERNET

October, 1993

Data transfer complete:

FIG. 5



Document Title: Open Market Payment

Document URL: http://payment.openmarket.com/ben/nph-payment

Open Market Payment

You have selected an item that requires payment

Merchant: Test Merchant
Description: Mead Data Central Article
Amount:2.85(US currency)

If you have an Open Market account click on "continue" below and you will be prompted for your account name and password. If you do not have an account, you can establish one on-line and return to this page to continue your purchase.

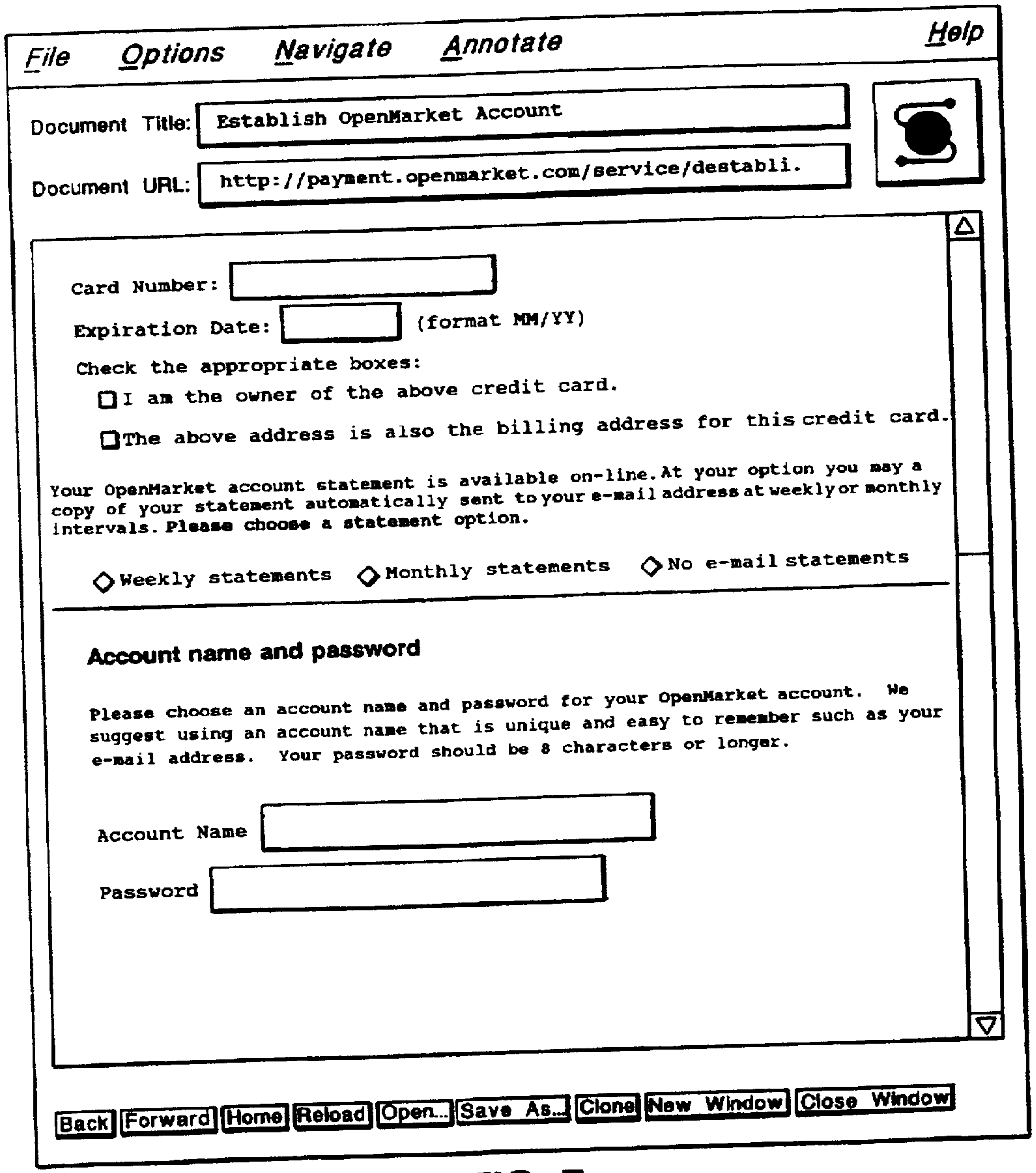
Open an account on-line

Continue with payment transaction.

NOTE:For demonstrations use the account name **testuser@openmarket.com** with the password **testuser**.

Open Market, Inc.

FIG. 6



Document Title: Establish OpenMarket Account

Document URL: http://payment.openmarket.com/service/destabli.

Card Number:

Expiration Date: (format MM/YY)

Check the appropriate boxes:

☐ I am the owner of the above credit card.

☐ The above address is also the billing address for this credit card.

Your OpenMarket account statement is available on-line. At your option you may a copy of your statement automatically sent to your e-mail address at weekly or monthly intervals. Please choose a statement option.

◇ Weekly statements ◇ Monthly statements ◇ No e-mail statements

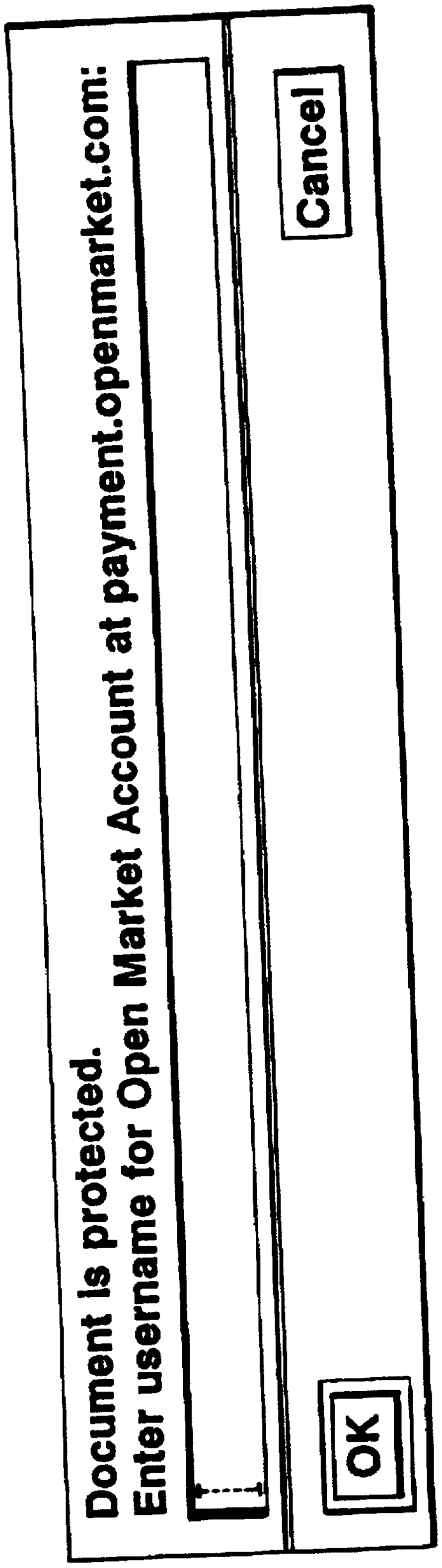
Account name and password

Please choose an account name and password for your OpenMarket account. We suggest using an account name that is unique and easy to remember such as your e-mail address. Your password should be 8 characters or longer.

Account Name

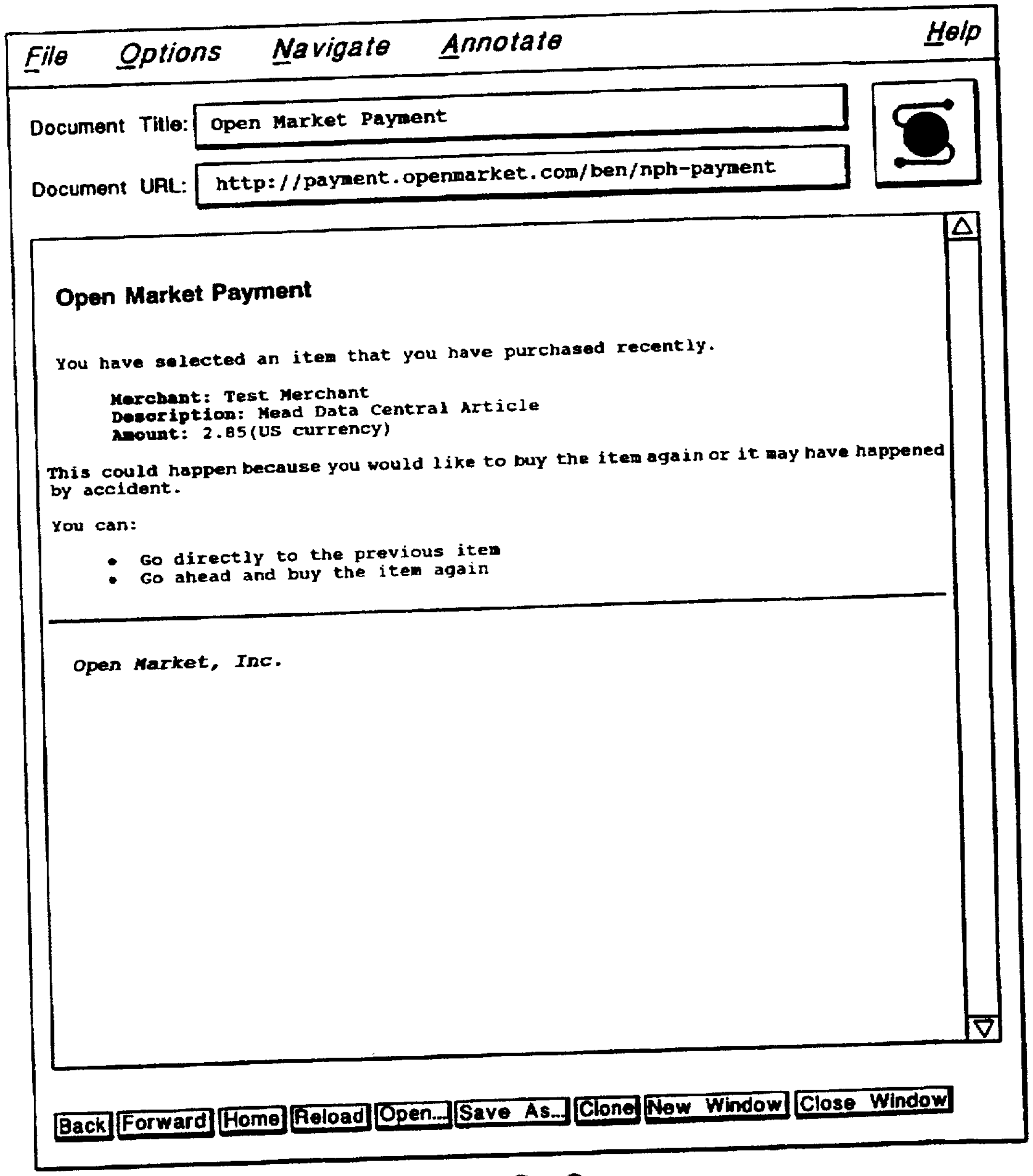
Password

FIG. 7



Document is protected.
Enter username for Open Market Account at payment.openmarket.com:

FIG. 8



Document Title: Open Market Payment

Document URL: http://payment.openmarket.com/ben/nph-payment

Open Market Payment

You have selected an item that you have purchased recently.

Merchant: Test Merchant
Description: Mead Data Central Article
Amount: 2.85(US currency)

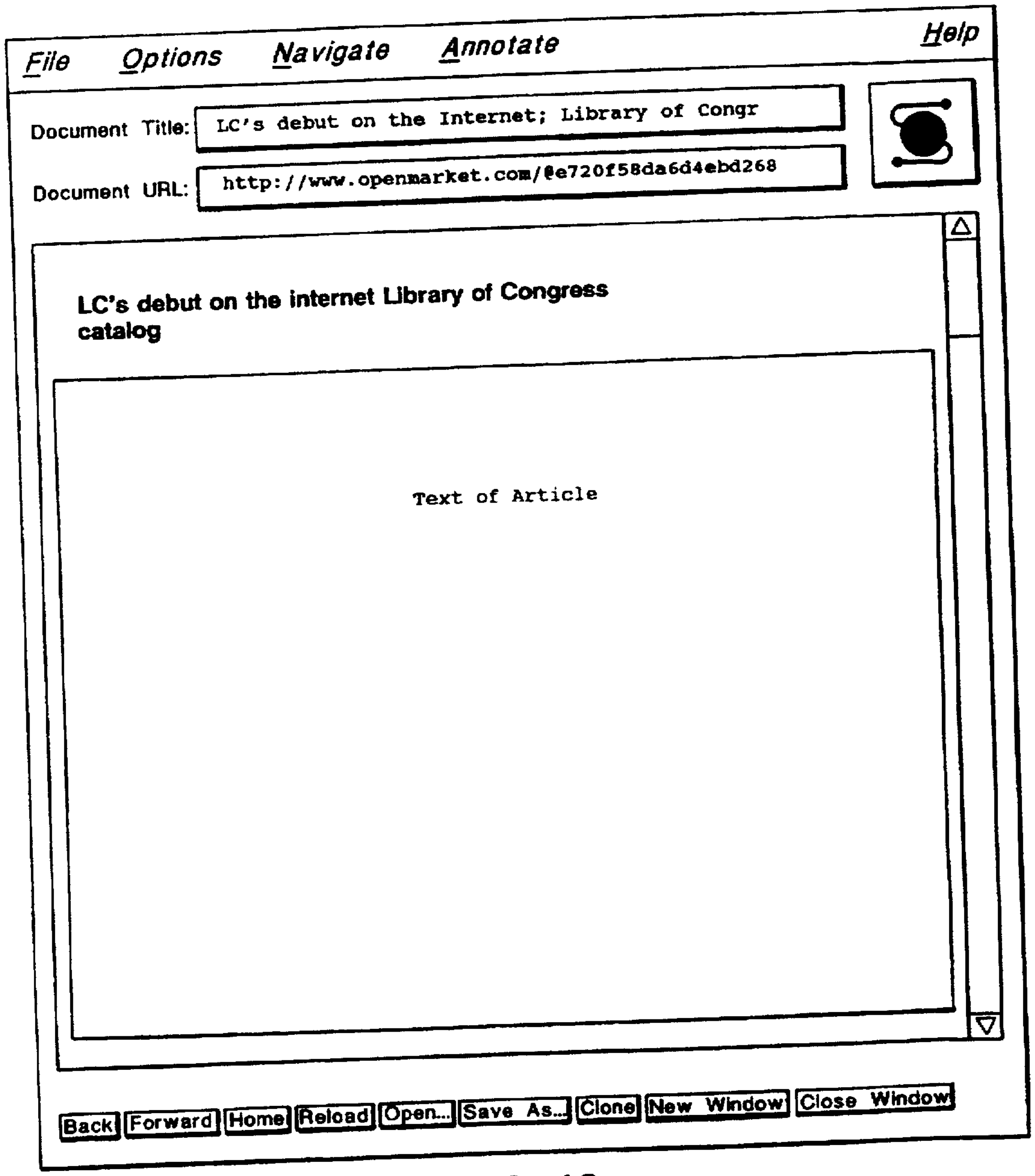
This could happen because you would like to buy the item again or it may have happened by accident.

You can:

- Go directly to the previous item
- Go ahead and buy the item again

Open Market, Inc.

FIG. 9



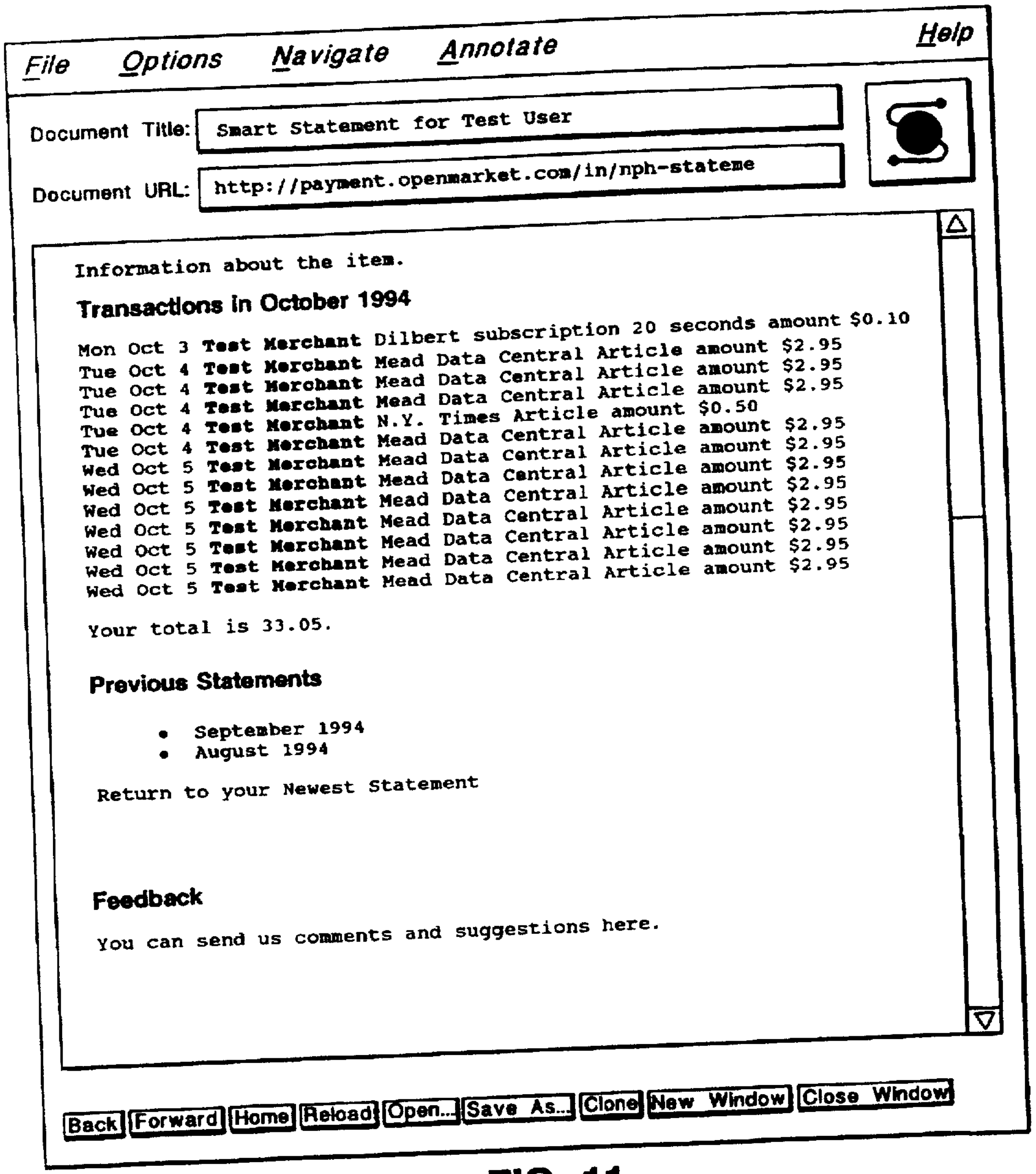
Document Title: LC's debut on the Internet; Library of Congr

Document URL: http://www.openmarket.com/@e720f58da6d4ebd268

LC's debut on the internet Library of Congress catalog

Text of Article

FIG. 10



Document Title: Smart Statement for Test User

Document URL: http://payment.openmarket.com/in/nph-stateme

Information about the item.

Transactions in October 1994

```
Mon Oct 3 Test Merchant Dilbert subscription 20 seconds amount $0.10
Tue Oct 4 Test Merchant Mead Data Central Article amount $2.95
Tue Oct 4 Test Merchant Mead Data Central Article amount $2.95
Tue Oct 4 Test Merchant Mead Data Central Article amount $2.95
Tue Oct 4 Test Merchant N.Y. Times Article amount $0.50
Tue Oct 4 Test Merchant Mead Data Central Article amount $2.95
Wed Oct 5 Test Merchant Mead Data Central Article amount $2.95
Wed Oct 5 Test Merchant Mead Data Central Article amount $2.95
Wed Oct 5 Test Merchant Mead Data Central Article amount $2.95
Wed Oct 5 Test Merchant Mead Data Central Article amount $2.95
Wed Oct 5 Test Merchant Mead Data Central Article amount $2.95
Wed Oct 5 Test Merchant Mead Data Central Article amount $2.95
Wed Oct 5 Test Merchant Mead Data Central Article amount $2.95
```

Your total is 33.05.

Previous Statements

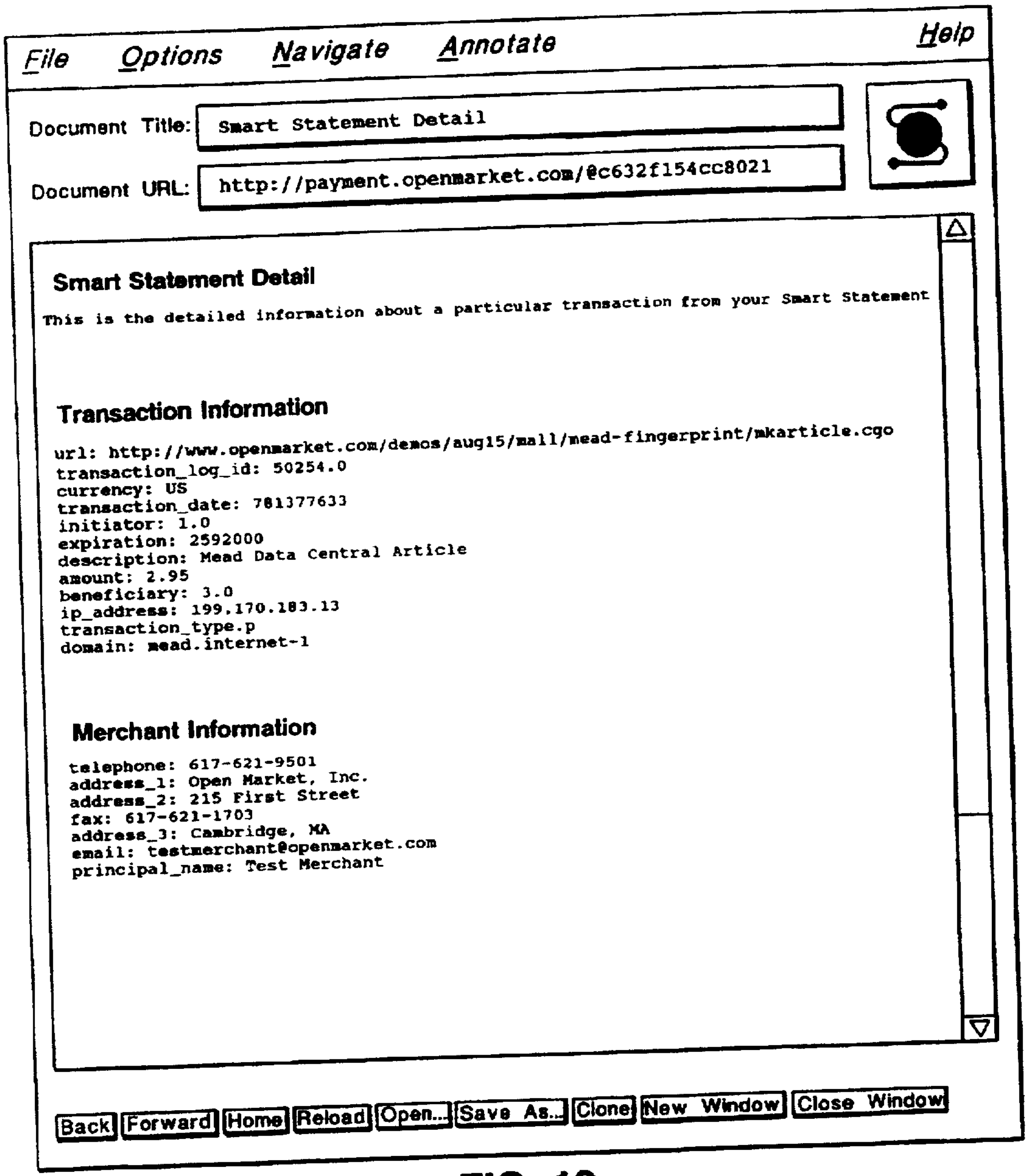
- September 1994
- August 1994

Return to your Newest Statement

Feedback

You can send us comments and suggestions here.

FIG. 11



Document Title: Smart Statement Detail

Document URL: http://payment.openmarket.com/@c632f154cc8021

Smart Statement Detail

This is the detailed information about a particular transaction from your Smart Statement

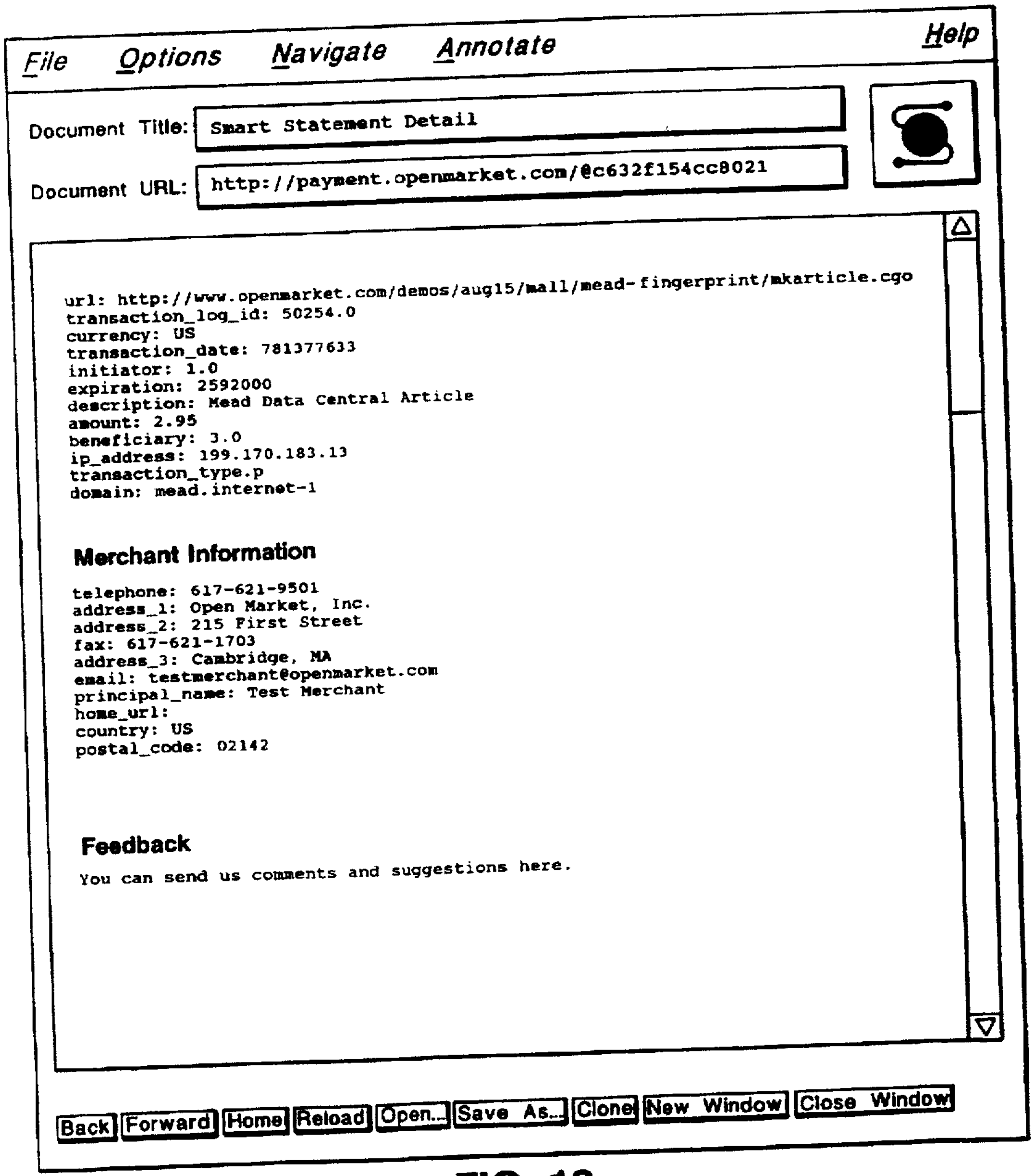
Transaction Information

```
url: http://www.openmarket.com/demos/aug15/mall/mead-fingerprint/mkarticle.cgo
transaction_log_id: 50254.0
currency: US
transaction_date: 781377633
initiator: 1.0
expiration: 2592000
description: Mead Data Central Article
amount: 2.95
beneficiary: 3.0
ip_address: 199.170.183.13
transaction_type.p
domain: mead.internet-1
```

Merchant Information

```
telephone: 617-621-9501
address_1: Open Market, Inc.
address_2: 215 First Street
fax: 617-621-1703
address_3: Cambridge, MA
email: testmerchant@openmarket.com
principal_name: Test Merchant
```

FIG. 12



Document Title: Smart Statement Detail

Document URL: http://payment.openmarket.com/@c632f154cc8021

```
url: http://www.openmarket.com/demos/aug15/mall/mead-fingerprint/mkarticle.cgo
transaction_log_id: 50254.0
currency: US
transaction_date: 781377633
initiator: 1.0
expiration: 2592000
description: Mead Data Central Article
amount: 2.95
beneficiary: 3.0
ip_address: 199.170.183.13
transaction_type.p
domain: mead.internet-1
```

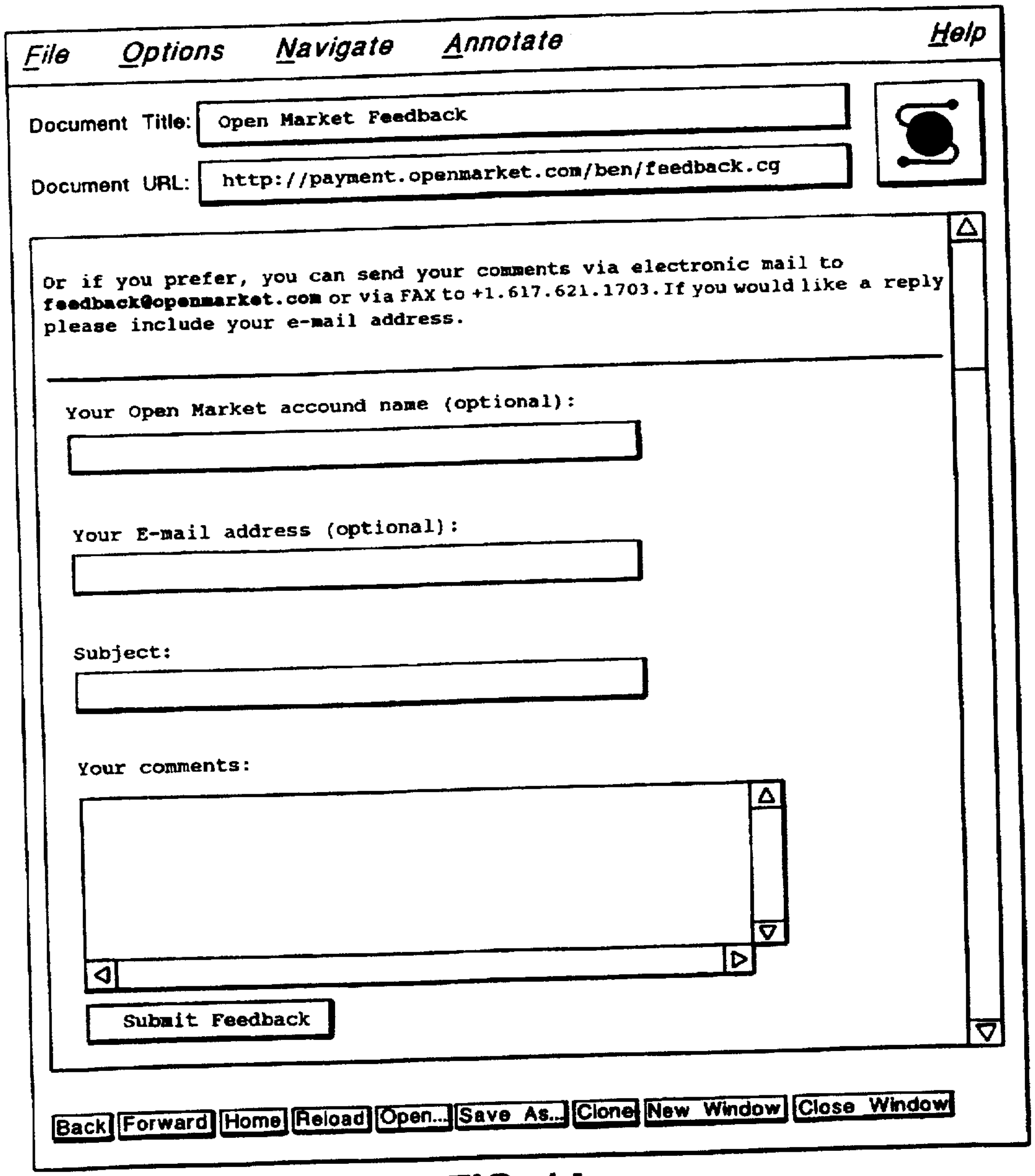
Merchant Information

```
telephone: 617-621-9501
address_1: Open Market, Inc.
address_2: 215 First Street
fax: 617-621-1703
address_3: Cambridge, MA
email: testmerchant@openmarket.com
principal_name: Test Merchant
home_url:
country: US
postal_code: 02142
```

Feedback

You can send us comments and suggestions here.

FIG. 13



Document Title: Open Market Feedback

Document URL: http://payment.openmarket.com/ben/feedback.cg

Or if you prefer, you can send your comments via electronic mail to **feedback@openmarket.com** or via FAX to +1.617.621.1703. If you would like a reply please include your e-mail address.

Your Open Market accound name (optional):

Your E-mail address (optional):

Subject:

Your comments:

FIG. 14

NETWORK SALES SYSTEM

REFERENCES TO APPENDICES

Microfiche appendices A–G, 4 sheets of 192 images total, are being submitted with the present application.

This invention relates to user-interactive network sales systems for implementing an open marketplace for goods or services over computer networks such as the Internet.

U.S. patent application Ser. No. 08/168,519, filed Dec. 16, 1993 by David K. Gifford and entitled "Digital Active Advertising," the entire disclosure of which is hereby incorporated herein in its entirety by reference, now abandoned, describes a network sales system that includes a plurality of buyer computers, a plurality of merchant computers, and a payment computer. A user at a buyer computer asks to have advertisements displayed, and the buyer computer requests advertisements from a merchant computer, which sends the advertisements to the buyer computer. The user then requests purchase of an advertised product, and the buyer computer sends a purchase message to the merchant computer. The merchant computer constructs a payment order that it sends to the payment computer, which authorizes the purchase and sends an authorization message to the merchant computer. When the merchant computer receives the authorization message it sends the product to the buyer computer.

The above-mentioned patent application also describes an alternative implementation of the network sales system in which, when the user requests purchase of an advertised product, the buyer computer sends a payment order directly to the payment computer, which sends an authorization message back to the buyer computer that includes an unforgeable certificate that the payment order is valid. The buyer computer then constructs a purchase message that includes the unforgeable certificate and sends it to the merchant computer. When the merchant computer receives the purchase request it sends the product to the buyer computer, based upon the pre-authorized payment order.

SUMMARY OF THE INVENTION

In one aspect, the invention provides a network-based sales system that includes at least one buyer computer for operation by a user desiring to buy a product, at least one merchant computer, and at least one payment computer. The buyer computer, the merchant computer, and the payment computer are interconnected by a computer network. The buyer computer is programmed to receive a user request for purchasing a product, and to cause a payment message to be sent to the payment computer that comprises a product identifier identifying the product. The payment computer is programmed to receive the payment message, to cause an access message to be created that comprises the product identifier and an access message authenticator based on a cryptographic key, and to cause the access message to be sent to the merchant computer. The merchant computer is programmed to receive the access message, to verify the access message authenticator to ensure that the access message authenticator was created using the cryptographic

key, and to cause the product to be sent to the user desiring to buy the product.

The invention provides a simple design architecture for the network sales system that allows the merchant computer to respond to payment orders from the buyer computer without the merchant computer having to communicate directly with the payment computer to ensure that the user is authorized to purchase the product and without the merchant computer having to store information in a database regarding which buyers are authorized to purchase which products. Rather, when the merchant computer receives an access message from the buyer computer identifying a product to be purchased, the merchant computer need only check the access message to ensure that it was created by the payment computer (thereby establishing for the merchant computer that the buyer is authorized to purchase the product), and then the merchant computer can cause the product to be sent to the buyer computer who has been authorized to purchase the product.

In another aspect, the invention features a network-based sales system that includes at least one buyer computer for operation by a user desiring to buy products, at least one shopping cart computer, and a shopping cart database connected to the shopping cart computer. The buyer computer and the shopping cart computer are interconnected by a computer network. The buyer computer is programmed to receive a plurality of requests from a user to add a plurality of respective products to a shopping cart in the shopping cart database, and, in response to the requests to add the products, to send a plurality of respective shopping cart messages to the shopping cart computer each of which includes a product identifier identifying one of the plurality of products. The shopping cart computer is programmed to receive the plurality of shopping cart messages, to modify the shopping cart in the shopping cart database to reflect the plurality of requests to add the plurality of products to the shopping cart, and to cause a payment message associated with the shopping cart to be created. The buyer computer is programmed to receive a request from the user to purchase the plurality of products added to the shopping cart and to cause the payment message to be activated to initiate a payment transaction for the plurality of products added to the shopping cart.

In another aspect, the invention features a network-based link message system that includes at least one client computer for operation by a client user and at least one server computer for operation by a server user. The client computer and the server computer are interconnected by a computer network. The client computer is programmed to send an initial link message to the server computer. The server computer is programmed to receive the initial link message from the client computer and to create, based on information contained in the initial link message, a session link message that encodes a state of interaction between the client computer and the server computer. The session link message includes a session link authenticator, computed by a cryptographic function of the session link contents, for authenticating the session link message. The server computer is programmed to cause the session link message to be sent to the client computer. The client computer is programmed to cause the session link message to be sent to a computer in the network that is programmed to authenticate the session link message by examining the session link authenticator and that is programmed to respond to the session link message based on the state of the interaction between the client computer and the server computer.

In another aspect, the invention features a network-based sales system that includes a merchant database having a plurality of digital advertisements and a plurality of respective product fulfillment items, at least one creation computer for creating the merchant database, and at least one merchant computer for causing the digital advertisements to be transmitted to a user and for causing advertised products to be transmitted to the user. The creation computer and the merchant computer are interconnected by a computer network. The creation computer is programmed to create the merchant database, and to transmit the digital advertisements and the product fulfillment items to the merchant computer. The merchant computer is programmed to receive the digital advertisements and product fulfillment items, to receive a request for a digital advertisement from a user, to cause the digital advertisement to be sent to the user, to receive from the user an access message identifying an advertised product, and to cause the product to be sent to the user in accordance with a product fulfillment item corresponding to the product.

In another aspect, the invention features a hypertext statement system that includes a client computer for operation by a client user and one or more server computers for operation by a server user. The client computer and the server computers are interconnected by a computer network. At least one of the server computers is programmed to record purchase transaction records in a database. Each of the purchase transaction records includes a product description. The server computer is programmed to transmit a statement document that includes the purchase transaction records to the client computer. The client computer is programmed to display the product descriptions, to receive a request from the client user to display a product corresponding to a product description displayed by the client computer, and to cause a product hypertext link derived from a purchase transaction record to be activated. At least one of the server computers is programmed to respond to activation of the product hypertext link by causing the product to be sent to the client computer.

In another aspect, the invention features a network payment system that includes at least one buyer computer for operation by a user desiring to buy a product and at least one payment computer for processing payment messages from the buyer computer. The buyer computer and the payment computer are interconnected by a computer network. The buyer computer is programmed to cause a payment message to be sent to the payment computer. The payment message includes a product identifier identifying the product that the user desires to buy. The payment computer is programmed to receive the payment message, to cause an access message to be created to enable the user to access the product, and to record a purchase transaction record in the settlement database. The buyer computer is programmed to cause a request for purchase transaction records to be sent to the payment computer. The payment computer is programmed to receive the request for purchase transaction records and to cause a document derived from the purchase transaction records to be sent to the buyer computer.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a network sales system in accordance with the present invention.

FIG. 2 (2-A through 2-I) is a flowchart diagram illustrating the operation of a purchase transaction in the network sales system of FIG. 1.

FIG. 3 (3-A through 3-B) is a flowchart diagram illustrating the use of a shopping cart for the purchase of products in connection with the network sales system of FIG. 1.

FIG. 4 (4-A through 4-C) is a flowchart diagram illustrating the operation of a smart statement in the network sales system of FIG. 1.

FIG. 5 is a screen snapshot of an advertising document that the merchant computer sends to the buyer computer in FIG. 2.

FIG. 6 is a screen snapshot of a confirmation document that the payment computer sends to the buyer computer in FIG. 2.

FIG. 7 is a screen snapshot of a new account document that the payment computer sends to the buyer computer in FIG. 2.

FIG. 8 is a screen snapshot of an account name prompt that the buyer computer creates in FIG. 2.

FIG. 9 is a screen snapshot of a document that the payment computer sends to the buyer computer in FIG. 2 and that provides an option either to repurchase or to use a previously purchased access.

FIG. 10 is a screen snapshot of a fulfillment document that the merchant computer sends to the buyer computer in FIG. 2.

FIG. 11 is a screen snapshot of a smart statement document that the payment computer sends to the buyer computer in FIG. 4.

FIGS. 12 and 13 are screen snapshots of a transaction detail document that the payment computer sends to the buyer computer in FIG. 4.

FIG. 14 is a screen snapshot of a customer service form that the payment computer sends to the buyer computer in FIG. 4.

DETAILED DESCRIPTION

With reference to FIG. 1, a network sales system in accordance with the present invention includes a buyer computer 12 operated by a user desiring to buy a product, a merchant computer 14, which may be operated by a merchant willing to sell products to the buyer or by a manager of the network sales system, a payment computer 16 typically operated by a manager of the network sales system, and a creation computer 20 typically operated by the merchant. The buyer, merchant, payment, and creation computers are all inter-connected by a computer network 10 such as the Internet.

Creation computer 20 is programmed to build a "store" of products for the merchant. A printout of a computer program for use in creating such a "store" in accordance with the present invention is provided as Appendix F.

The products advertised by merchant computer 14 may be, for example, newspaper or newsletter articles available for purchase by buyers. Creation computer 20 creates a digital advertisement database 18 that stores advertising documents (which may for example be in the form of summaries of newspaper or newsletter articles, accompanied by prices) and product fulfillment items (which may be the products themselves if the products can be transmitted over the network, or which may be hard goods identifiers if the products are hard goods, i.e., durable products as opposed to information products). Creation computer 20 transmits contents of the advertising document database 18 to merchant computer 14 to enable the merchant computer to cause advertisements and products to be sent to buyers. Merchant computer 14 maintains advertising documents locally in advertising document database 15. In an alternative embodiment, the creation computer does not have a local digital advertisement database, but instead updates a remote advertising document database on a merchant computer. These updates can be accomplished using HTML forms or other remote database technologies as is understood by practitioners of the art.

Payment computer **16** has access to a settlement database **22** in which payment computer **16** can record details of purchase transactions. The products may be organized into various "domains" of products, and payment computer **16** can access settlement database **22** to record and retrieve records of purchases of products falling within the various domains. Payment computer **16** also has access to a shopping cart database **21** in which a "shopping cart" of products that a user wishes to purchase can be maintained as the user shops prior to actual purchase of the contents of the shopping cart.

With reference to FIG. **2**, a purchase transaction begins when a user at buyer computer **12** requests advertisements (step **24**) and buyer computer **12** accordingly sends an advertising document URL (universal resource locator) to merchant computer **14** (step **26**). The merchant computer fetches an advertising document from the advertising document database (step **28**) and sends it to the buyer computer (step **30**). An example of an advertising document is shown in FIG. **5**. Details of URLs and how they are used are found in the microfiche Appendix G.

The user browses through the advertising document and eventually requests a product (step **32**). This results in the buyer computer sending payment URL A to the payment computer (step **34**). Payment URL A includes a product identifier that represents the product the user wishes to buy, a domain identifier that represents a domain of products to which the desired product belongs, a payment amount that represents the price of the product, a merchant computer identifier that represents merchant computer **14**, a merchant account identifier that represents the particular merchant account to be credited with the payment amount, a duration time that represents the length of time for which access to the product is to be granted to the user after completion of the purchase transaction, an expiration time that represents a deadline beyond which this particular payment URL cannot be used, a buyer network address, and a payment URL authenticator that is a digital signature based on a cryptographic key. The payment URL authenticator is a hash of other information in the payment URL, the hash being defined by a key shared by the merchant and the operator of the payment computer.

In an alternative embodiment, step **34** consists of the buyer computer sending a purchase product message to the merchant computer, and the merchant computer provides payment VRL A to the buyer computer in response to the purchase product message. In this alternative embodiment, payment URL A contains the same contents as above. The buyer computer then sends the payment URL A it has received from the merchant computer to the payment computer.

When the payment computer receives the payment URL it verifies whether the payment URL authenticator was created from the contents of the payment URL using the cryptographic key (step **36**). If not, the payment computer sends a document to the buyer computer indicating that access to the network sales system is denied (step **38**). Otherwise, the payment computer determines whether the expiration time has past (step **40**). If it has, the payment computer sends a document to the buyer computer indicating that the time has expired (step **41**). Otherwise, the payment computer checks the buyer computer network

address to see if it matches the one specified in the payment URL (step **42**). If it does not match, the payment computer sends a document to the buyer computer indicating that access to the network payment system is denied (step **43**). Otherwise, the payment computer sends a payment confirmation document to the buyer computer, the payment confirmation document including an "open" link and a "continue" link (step **44**).

An example of a confirmation document is shown in FIG. **6**. The confirmation document asks the user to click on a "continue" button if the user already has an account with the payment computer, or to click on an "open" button if the user does not already have an account and wishes to open one.

If the user clicks on the "open" button (step **46**), the buyer computer sends payment URL C to the payment computer (step **48**), payment URL C being similar to payment URL A but also indicating that the user does not yet have an account. The payment computer creates a new account document (step **50**) and sends it to the buyer computer (step **52**). An example of a new account document is shown in FIG. **7**. When the user receives the new account document he enters the new account name, an account password, a credit card number, the credit card expiration date, and security information such as the maiden name of the user's mother (step **54**), and presses a "submit" button (not shown in FIG. **7**). The buyer computer sends the new account information to the payment computer (step **56**), which enters the new account in the settlement database (step **58**).

If the user clicks on the "continue" button (step **60**), the buyer computer sends payment URL B to the payment computer (step **62**), payment URL B being similar to payment URL A but also indicating that the user already has an account. The payment computer then instructs the buyer computer to provide the account name and password (steps **64** and **66**), and the buyer computer prompts the user for this information by creating an account name prompt (example shown in FIG. **8**) and a similar password prompt. The user enters the information (step **68**) and the buyer computer sends the account name and password to the payment computer (step **70**).

The payment computer verifies whether the user name and password are correct (step **72**). If they are not correct, the payment computer sends a document to the buyer computer indicating that access to the network sales system is denied (step **74**). Otherwise, the payment computer determines whether additional security is warranted, based on, e.g., whether the payment amount exceeds a threshold (step **73**). If additional security is warranted, the payment computer creates a challenge form document and sends it to the buyer computer (step **75**). The user enters the security information (step **77**), the buyer computer sends the security information to the payment computer (step **79**), and the payment computer determines whether the security information is correct (step **81**). If it is not correct, the payment computer sends a document to the buyer computer indicating that access to the network sales system is denied (step **83**).

If the security information is correct, or if additional security was not warranted, the payment computer checks the settlement database to determine whether the user has unexpired access to the domain identifier contained in the payment URL (step **82**). If so, the payment computer sends to the buyer computer a document providing an option either to repurchase or to use the previously purchased access (step **84**). An example of such a document is shown in FIG. **9**. The user can respond to the recent purchase query document by choosing to access the previously purchased document (step **85**) or to go ahead and buy the currently selected product (step **86**).

If the user chooses to access the previously purchased document, the buyer computer skips to step **92** (see below). If the user chooses to buy the currently selected product, the payment computer calculates an actual payment amount that may differ from the payment amount contained in the payment URL (step **87**). For example, the purchase of a product in a certain domain may entitle the user to access other products in the domain for free or for a reduced price for a given period of time.

The payment computer then verifies whether the user account has sufficient funds or credit (step **76**). If not, the payment computer sends a document to the buyer computer indicating that the user account has insufficient funds (step **78**). Otherwise, the payment computer creates an access URL (step **80**) that includes a merchant computer identifier, a domain identifier, a product identifier, an indication of the end of the duration time for which access to the product is to be granted, the buyer network address, and an access URL authenticator that is a digital signature based on a cryptographic key. The access URL authenticator is a hash of other information in the access URL, the hash being defined by a key shared by the merchant and the operator of the payment computer. The payment computer then records the product identifier, the domain, the user account, the merchant account, the end of duration time, and the actual payment amount in the settlement database (step **88**).

The payment computer then sends a redirect to access URL to the buyer computer (step **90**), which sends the access URL to the merchant computer (step **92**). The merchant computer verifies whether the access URL authenticator was created from the contents of the access URL using the cryptographic key (step **94**). If not, the merchant computer sends a document to the buyer computer indicating that access to the product is denied (step **96**).

Otherwise, the merchant computer verifies whether the duration time for access to the product has expired (step **98**). This is done because the buyer computer can request access to a purchased product repeatedly. If the duration time has expired, the merchant computer sends a document to the buyer computer indicating that the time has expired (step **100**). Otherwise the merchant computer verifies that the buyer computer network address is the same as the buyer network address in the access URL (step **101**), and if so, sends a fulfillment document to the buyer computer (step **102**), which is displayed by the buyer computer (step **104**). An example of a fulfillment document is shown in FIG. **10**. Otherwise, the merchant computer sends a document to the buyer computer indicating that access is not allowed (step **103**).

With reference now to FIG. **3**, when the merchant computer sends the advertising document to the buyer computer, the user may request that a product be added to a shopping cart in the shopping cart database rather than request that the product be purchased immediately. The buyer computer sends a shopping cart URL to the payment computer (step **108**), the shopping cart URL including a product identifier, a domain identifier, a payment amount, a merchant computer identifier, a merchant account identifier, a duration time, an expiration time, and a shopping cart URL authenticator that is a digital signature based on a cryptographic key. The shopping cart URL authenticator is a hash of other information in the shopping cart URL, the hash being defined by a key shared by the merchant and the operator of the payment computer.

The payment computer verifies whether the shopping cart URL authenticator was created from the contents of the shopping cart URL using a cryptographic key (step **110**). If not, the payment computer sends a document to the buyer computer indicating that access to the network sales system is denied (step **112**). Otherwise, before any modification to a user's shopping cart is allowed, user authentication is performed (step **113**) in a manner analogous to steps **40–81**. Once the user is authenticated, the payment computer creates or updates a payment URL for the shopping cart (step **114**).

The user then either requests more advertisements (step **24** in FIG. **2**) and possibly adds another product to the shopping cart, requests display of the shopping cart (step **116**), or requests purchase of the entire contents of the shopping cart (step **124**). If the user requests display of the shopping cart (step **116**), the buyer computer sends a fetch shopping cart request to the payment computer (step **118**), and the payment computer and buyer computer (step **119**) perform steps analogous to steps **64–81**. The payment computer returns the contents of the shopping cart to the buyer computer (step **120**), which displays the contents of the shopping cart (step **122**). If the user requests that the entire contents of the shopping cart be purchased (step **124**) the buyer computer causes the payment URL for the shopping cart to be activated (step **126**) and the payment URL is processed in a manner analogous to the processing of payment URLs for individual products (beginning with step **36** in FIG. **2**).

With reference now to FIG. **4**, a user can request display of a "smart statement" that lists purchase transactions for a given month (step **128**). When the buyer computer receives such a request, it sends a smart statement URL to the payment computer (step **130**).

When the payment computer receives the smart statement URL, it verifies whether the smart statement URL authenticator was created from the contents of the smart statement URL using a cryptographic key (step **132**). If not, the payment computer sends a document to the buyer computer indicating that access is denied (step **134**). Otherwise, the payment computer checks to determine whether the buyer network address in the smart statement URL matches the buyer computer's actual network address (step **136**). If not, the payment computer sends a document to the buyer computer indicating that access is denied (step **138**). Otherwise (step **140**), the payment computer and buyer computer perform a set of steps analogous to steps **64–81** in FIG. **2** (payment computer requests account name and password, user provides the requested information, and payment computer verifies the information).

In an alternative embodiment steps **132–138** are omitted.

After verification of account information is complete, the payment computer retrieves the requested settlement data from the settlement database, creates a smart statement document for the buyer, and sends the smart statement document to the buyer computer (step **142**). An example of a smart statement document is shown in FIG. **11**. Each purchase transaction record in the smart statement document includes the data of the transaction, the name of the merchant, an identification of the product, and the payment amount for the product. The smart statement document also includes a transaction detail URL for each purchase transaction (these URLs, or hypertext links, are discussed below and are not shown in FIG. **11**). The smart statement document also identifies previous statements that the user may wish to have displayed.

The buyer computer displays the retrieved document (step 144), and the user may request transaction details for a particular transaction listed on the smart statement (step 146). If so, the buyer computer sends a transaction detail URL (or "payment detail URL") to the payment computer (step 148). The transaction detail URL includes a transaction identifier, a buyer network address, and a transaction detail URL authenticator. When the payment computer receives the transaction detail URL, it performs (step 150) a set of steps analogous to steps 132–140 (verification of URL authenticator, buyer network address, and account information). The payment computer then retrieves from the settlement database data corresponding to the payment transaction specified in the transaction detail URL, creates a transaction detail document, and sends it to the buyer computer (step 152).

An example of a transaction detail document is shown in FIGS. 12 and 13. The document displays a number of items of information about the transaction, including the transaction date, end of the duration time ("expiration"), a description of the product, the payment amount, the domain corresponding to the product, an identification of the merchant, and the merchant's address.

The smart statement document and the transaction detail document both include customer service URLs (hypertext links) that allow the user to request customer service (i.e., to send comments and suggestions to the payment computer). When the user requests customer service (step 154), the buyer computer sends the customer service URL to the payment computer (step 156), which creates a customer service form and sends it to the buyer computer (step 158). An example of a customer service form is shown in FIG. 14. The user types comments into the customer service form (step 160), and the buyer computer sends the user's comments to the payment computer (step 162). The payment computer then posts the user comments and sends a thank you document to the buyer computer (step 164).

A user may request display of a product included in the smart statement. When the user requests that the product be displayed (step 166), the buyer computer sends the access URL contained in the smart statement document to the merchant computer (step 168), and the buyer computer and merchant computer perform a set of steps analogous to steps 94–104 in FIG. 2 (authentication of access URL, verification whether duration time has expired, verification of buyer network address, and transmission of fulfillment document to buyer computer).

Whenever the present application states that one computer sends a URL to another computer, it should be understood that in preferred embodiments the URL is sent in a standard HTTP request message, unless a URL message is specified as a redirection in the present application. The request message includes components of the URL as described by the standard HTTP protocol definition. These URL components in the request message allow the server to provide a response appropriate to the URL. The term "URL" as used the present application is an example of a "link," which is a pointer to another document or form (including multimedia documents, hypertext documents including other links, or audio/video documents).

When the present application states that one computer sends a document to another computer, it should be understood that in preferred embodiments the document is a success HTTP response message with the document in the body of the message. When the present application states that a server sends an account name and password request message to the client, it should be understood that in preferred embodiments the account name and password request message is an unauthorized HTTP response. A client computer sends account name and password information to a server as part of a request message with an authorization field.

The software architecture underlying the particular preferred embodiment is based upon the hypertext conventions of the World Wide Web. Appendix A describes the Hypertext Markup Language (HTML) document format used to represent digital advertisements. Appendix B describes the HTML forms fill out support in Mosaic 2.0. Appendix C is a description of the Hypertext Transfer Protocol (HTTP) between buyer and merchant computers. Appendix D describes how documents are named with Uniform Resource Locators (URLs) in the network of computers, and Appendix E describes the authentication of URLs using digital signatures.

A printout of a computer program for use in creating and operating such a "store" in accordance with the present invention is provided as Appendix F. A printout of a computer program for use in operating other aspects of the network sales system in accordance with the present invention is provided in the microfiche appendix G.

There has been described a new and useful network-based sales system. It is apparent that those skilled in the art may make numerous modifications and departures from the specific embodiments described herein without departing from the spirit and scope of the claimed invention.

What is claimed is:

1. A network-based sales system, comprising:

at least one buyer computer for operation by a user desiring to buy a product;

at least one merchant computer; and

at least one payment computer;

said buyer computer, said merchant computer, and said payment computer being interconnected by a computer network;

said buyer computer being programmed to receive a user request for purchasing a product, and to cause a payment message to be sent to said payment computer that comprises a product identifier identifying said product;

said payment computer being programmed to receive said payment message, to cause an access message to be created that comprises said product identifier and an access message authenticator based on a cryptographic key, and to cause said access message to be sent to said merchant computer; and

said merchant computer being programmed to receive said access message, to verify said access message authenticator to ensure that said access message authenticator was created using said cryptographic key, and to cause said product to be sent to said user desiring to buy said product.

2. A network-based sales system in accordance with claim 1, wherein said payment message and said access message each comprises a universal resource locator.

3. A network-based sales system in accordance with claim 1, wherein said payment computer is programmed to identify said merchant computer upon receipt of said payment message from said buyer computer.

4. A network-based sales system in accordance with claim 1, wherein said access message comprises a buyer network address.

5. A network-based sales system in accordance with claim 4, wherein:

said product can be transmitted from one computer to another; and

said merchant computer causes said product to be sent to said user by transmitting said product to said buyer network address only.

6. A network-based sales system in accordance with claim 4, wherein said merchant computer is programmed to verify whether said buyer network address in said access message matches the actual network address of said buyer computer.

7. A network-based sales system in accordance with claim 1, wherein said payment message comprises a buyer network address.

8. A network-based sales system in accordance with claim 7, wherein said payment computer is programmed to verify whether said buyer network address in said payment message matches the actual network address of said buyer computer.

9. A network-based sales system in accordance with claim 1, wherein said access message authenticator comprises a cryptographic function of contents of said access message based on said cryptographic key.

10. A network-based sales system in accordance with claim 1, wherein said payment computer is programmed to verify said payment message authenticator to ensure that said payment message authenticator was created using said cryptographic key.

11. A network-based sales system in accordance with claim 10, wherein said payment message authenticator comprises a cryptographic function of contents of said payment message based on said cryptographic key.

12. A network-based sales system in accordance with claim 1, wherein said payment message comprises a payment amount.

13. A network-based sales system in accordance with claim 1, wherein said payment message comprises a merchant account identifier that identifies a merchant account.

14. A network-based sales system in accordance with claim 1, wherein said buyer computer is programmed to transmit a user account identifier to said payment computer that identifies a user account.

15. A network-based sales system in accordance with claim 14, wherein:

said payment message comprises a payment amount; and

said payment computer is programmed to ensure that said user account has sufficient funds or credit to cover said payment amount.

16. A network-based sales system in accordance with claim 14, wherein:

said payment message comprises a payment amount and a merchant account identifier that identifies a merchant account; and

said payment computer is programmed to record said payment amount, said user account, and said merchant account in a settlement database.

17. A network-based sales system in accordance with claim 16, wherein:

said payment message comprises a domain identifier; and

said payment computer is programmed to record said domain identifier and said user account in a settlement database.

18. A network-based sales system in accordance with claim 17, wherein said payment computer is programmed to check said settlement database, upon receipt of said payment message, to determine whether said user account has previously purchased a product associated with said domain identifier.

19. A network-based sales system in accordance with claim 18, wherein said payment computer is programmed to determine an actual payment amount for said product identified by said product identifier in said payment message based on whether said user account has previously purchased a product associated with said domain identifier.

20. A network-based sales system in accordance with claim 1, wherein said buyer computer is programmed to transmit a user authenticator to said payment computer and said payment computer is programmed to verify said user authenticator.

21. A network-based sales system in accordance with claim 20, wherein said user authenticator comprises a password.

22. A network-based sales system in accordance with claim 20, wherein:

said buyer computer is programmed to transmit security information to said payment computer;

said payment computer is programmed to transmit a challenge form to said buyer computer under a predetermined condition, said challenge form asking for said security information previously transmitted by said buyer computer to said payment computer;

said payment computer is programmed to respond to said challenge form by querying said user for said security information and transmitting said security information to said payment computer; and

said payment computer is programmed to verify authenticity of said security information.

23. A network-based sales system in accordance with claim 22, wherein:

said payment message comprises a payment amount; and

said predetermined condition comprises receipt of a payment amount in said payment message that exceeds a threshold.

24. A network-based sales system in accordance with claim 1, wherein said payment message comprises a merchant computer identifier that identifies said merchant computer.

25. A network-based sales system in accordance with claim 24, wherein said access message comprises said merchant computer identifier.

26. A network-based sales system in accordance with claim 1, wherein said payment message comprises a duration time that specifies a length of time for which access to said product is to be granted.

27. A network-based sales system in accordance with claim 26, wherein said payment computer is programmed to use said duration time to compute an end of duration time and to cause said end of duration time to be included in said access message.

28. A network-based sales system in accordance with claim 27, wherein said merchant computer is programmed to verify, upon receipt of said access message, that said end of duration time has not past.

29. A network-based sales system in accordance with claim 1, wherein said payment message comprises an expiration time after which said payment message can no longer be used.

30. A network-based sales system in accordance with claim 29, wherein said payment computer is programmed to verify, upon receipt of said payment message, that said expiration time has not past.

31. A network-based sales system in accordance with claim 1, wherein:

said payment computer is programmed to cause said access message to be sent to said buyer computer; and said buyer computer is programmed to cause said access message received from said payment computer to be sent to said merchant computer.

32. A network-based sales system, comprising:

at least one buyer computer for operation by a user desiring to buy a product;

at least one merchant computer; and

at least one payment computer;

said buyer computer, said merchant computer, and said payment computer being interconnected by a computer network;

said buyer computer being programmed to receive a user request for purchasing a product, and to cause a payment URL to be sent to said payment computer that comprises a product identifier identifying said product, a payment amount, and a payment URL authenticator comprising a cryptographic function of contents of said payment URL based on a cryptographic key;

said payment computer being programmed to receive said payment URL, to verify said payment URL authenticator to ensure that said payment URL authenticator was created using said cryptographic key, to ensure that said user has sufficient funds or credit to cover said payment amount, to identify said merchant computer operated by said merchant willing to sell said product to said buyer, to cause an access URL to be created that comprises said product identifier and an access URL authenticator comprising a cryptographic function of contents of said access URL based on a cryptographic key, and to cause said access URL to be sent to said buyer computer;

said buyer computer being programmed to cause said access URL received from said payment computer to be sent to said merchant computer; and

said merchant computer being programmed to receive said access URL, to verify said access URL authenticator to ensure that said access URL authenticator was created using said cryptographic key, and to cause said product to be sent to said user desiring to buy said product.

33. A method of operating a payment computer in a computer network comprising at least one buyer computer for operation by a user desiring to buy a product, at least one merchant computer, and at least one payment computer, the method comprising the steps of:

receiving, at said payment computer, a payment message that said buyer computer has caused to be sent to said payment computer in response to a user request for purchasing a product, said payment message comprising a product identifier identifying said product;

causing an access message to be created that comprises said product identifier and an access message authenticator based on a cryptographic key; and

causing said access message to be sent to said merchant computer, said merchant computer being programmed to receive said access message, to verify said access message authenticator to ensure that said access message authenticator was created using said cryptographic key, and to cause said product to be sent to said user desiring to buy said product.

34. A network-based sales system, comprising:

at least one buyer computer for operation by a user desiring to buy products;

at least one shopping cart computer; and

a shopping cart database connected to said shopping cart computer;

said buyer computer and said shopping cart computer being interconnected by a computer network;

said buyer computer being programmed to receive a plurality of requests from a user to add a plurality of respective products to a shopping cart in said shopping cart database, and, in response to said requests to add said products, to send a plurality of respective shopping cart messages to said shopping cart computer each of which comprises a product identifier identifying one of said plurality of products;

said shopping cart computer being programmed to receive said plurality of shopping cart messages, to modify said shopping cart in said shopping cart database to reflect said plurality of requests to add said plurality of products to said shopping cart, and to cause a payment message associated with said shopping cart to be created; and

said buyer computer being programmed to receive a request from said user to purchase said plurality of products added to said shopping cart and to cause said payment message to be activated to initiate a payment transaction for said plurality of products added to said shopping cart;

said shopping cart being a stored representation of a collection of products, said shopping cart database being a database of stored representations of collections of products, and said shopping cart computer being a computer that modifies said stored representations of collections of products in said database.

35. A network-based sales system in accordance with claim 34, wherein said shopping cart computer is programmed to cause said payment message to be created before said buyer computer causes said payment message to be activated.

36. A network-based sales system in accordance with claim 34, wherein said buyer computer is programmed to receive a request from said user to display said plurality of products added to said shopping cart.

37. A network-based sales system in accordance with claim 36, wherein said buyer computer is programmed to transmit a fetch shopping cart request to said payment computer in response to receipt of said request from said user.

38. A network-based sales system in accordance with claim 37, wherein:

said payment computer is programmed to respond to said fetch shopping cart request by transmitting a message to said buyer computer indicating said plurality of products added to said shopping cart; and

said buyer computer is programmed to display said plurality of products added to said shopping cart.

39. A method of operating a shopping cart computer in a computer network comprising at least one buyer computer for operation by a user desiring to buy products, at least one shopping cart computer, and a shopping cart database connected to said shopping cart computer, said method comprising the steps of:

receiving, at said shopping cart computer, a plurality of shopping cart messages sent to said shopping cart computer by said buyer computer in response to receipt of a plurality of requests from a user to add a plurality of respective products to a shopping cart in said shopping cart database, each of said shopping cart messages comprising a product identifier identifying one of said plurality of products;

modifying said shopping cart in said shopping cart database to reflect said plurality of requests to add said plurality of products to said shopping cart; and causing a payment message associated with said shopping cart to be created;

said buyer computer being programmed to receive a request from said user to purchase said plurality of products added to said shopping cart and to cause said payment message to be activated to initiate a payment transaction for said plurality of products added to said shopping cart;

said shopping cart being a stored representation of a collection of products, said shopping cart database being a database of stored representations of collections of products, and said shopping cart computer being a computer that modifies said stored representations of collections of products in said database.

40. A network-based link message system, comprising:

at least one client computer for operation by a client user; and

at least one server computer for operation by a server user;

said client computer and said server computer being interconnected by a computer network;

said client computer being programmed to send an initial link message to said server computer;

said server computer being programmed to receive said initial link message from said client computer, to create, based on information contained in said initial link message, a session link message that encodes a state of interaction between said client computer and said server computer, said session link message comprising a session link authenticator, computed by a cryptographic function of said session link contents, for authenticating said session link message, and to cause said session link message to be sent to said client computer;

said client computer being programmed to cause said session link message to be sent to a computer in said network that is programmed to authenticate said session link message by examining said session link authenticator and that is programmed to respond to said session link message based on said state of said interaction between said client computer and said server computer.

41. A network-based link message system in accordance with claim **40**, wherein:

said client computer comprises a buyer computer for operation by a user desiring to buy a product;

said server computer comprises a payment computer for operation by a manager of said network-based link message system; and

said network-based link message system further comprises a merchant computer for operation by a merchant willing to sell said product to said buyer.

42. A network-based link message system in accordance with claim **41**, wherein said computer that is programmed to authenticate said session link message comprises said merchant computer.

43. A network-based link message system in accordance with claim **41**, wherein said initial link message comprises a payment message to said payment computer that comprises a product identifier identifying said product.

44. A network-based link message system in accordance with claim **43**, wherein said session link message comprises an access message that comprises said product identifier to be created.

45. A network-based link message system in accordance with claim **44**, wherein said merchant computer is programmed to respond to said access message by causing said product to be sent to said user desiring to buy said product.

46. A network-based link message system in accordance with claim **40**, wherein said initial link message and said session link message comprise universal resource locators.

47. A network-based link message system in accordance with claim **40**, wherein:

said session link authenticator comprises a cryptographic function of contents of said session link message based on a cryptographic key; and

said computer to which said client computer is programmed to cause said session link message to be sent is programmed to verify that said session link authenticator was created using said cryptographic key.

48. A method of operating a server computer in a network-based link message system comprising at least one client computer for operation by a client user and at least one server computer for operation by a server user, said client computer and said server computer being interconnected by a computer network, said method comprising the steps of:

receiving, at said server computer, an initial link message sent to said server computer by said client computer;

creating, based on information contained in said initial link message, a session link message that encodes a state of interaction between said client computer and said server computer, said session link message comprising a session link authenticator, computed by a cryptographic function of said session link contents, for authenticating said session link message; and

causing said session link message to be sent to said client computer;

said client computer being programmed to cause said session link message to be sent to a computer in said network that is programmed to authenticate said session link message by examining said session link authenticator and that is programmed to respond to said session link message based on said state of said interaction between said client computer and said server computer.

* * * * *

Exhibit "C"

US005909492A

United States Patent [19]

Payne et al.

[11] **Patent Number: 5,909,492**

[45] **Date of Patent: Jun. 1, 1999**

[54] **NETWORK SALES SYSTEM**

[75] Inventors: **Andrew C. Payne**, Lincoln; **Lawrence C. Stewart**, Burlington, both of Mass.; **David J. Mackie**, Brookdale, Calif.

[73] Assignee: **Open Market, Incorporated**, Cambridge, Mass.

[21] Appl. No.: **08/878,396**

[22] Filed: **Jun. 18, 1997**

Related U.S. Application Data

[63] Continuation of application No. 08/328,133, Oct. 24, 1994, Pat. No. 5,715,314.

[51] **Int. Cl.**^{6} **H04L 9/00**

[52] **U.S. Cl.** **380/24**; 380/23; 380/25; 380/49; 380/50; 705/26; 705/27; 705/39; 705/40; 705/44

[58] **Field of Search** 380/4, 9, 21, 23, 380/24, 25, 49, 50; 235/379, 380; 705/26, 27, 39, 40, 41, 42, 43, 44, 14, 16

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,305,059	12/1981	Benton .	
4,528,643	7/1985	Freeny, Jr. .	
4,529,870	7/1985	Chaum	235/380
4,578,530	3/1986	Zeidler .	
4,734,858	3/1988	Schlafly .	
4,755,940	7/1988	Brachtl et al. .	
4,759,063	7/1988	Chaum	380/30
4,759,064	7/1988	Chaum	380/30
4,775,935	10/1988	Yourick .	
4,795,890	1/1989	Goldman	235/380
4,799,156	1/1989	Shavit et al. .	
4,812,628	3/1989	Boston et al.	235/380
4,827,508	5/1989	Shear	380/4
4,891,503	1/1990	Jewel	235/380
4,922,521	5/1990	Krikke et al. .	
4,926,480	5/1990	Chaum	380/23
4,935,870	6/1990	Burk, Jr. et al. .	
4,947,028	8/1990	Gorog .	
4,947,430	8/1990	Chaum	380/25
4,949,380	8/1990	Chaum	380/30
4,972,318	11/1990	Brown et al.	705/26
4,977,595	12/1990	Ohta et al.	380/24
4,982,346	1/1991	Girouard et al. .	
4,987,593	1/1991	Chaum	380/3
4,991,210	2/1991	Chaum	380/30
4,992,940	2/1991	Dworkin .	

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

0 172 670	2/1986	European Pat. Off.	G07F 7/00
0-542-298-A2	5/1993	European Pat. Off. .	
4-10191	1/1992	Japan	705/26
2102606	2/1983	United Kingdom .	
WO 91/16691	10/1991	WIPO .	
WO 93/10503	5/1993	WIPO	G06F 15/30

OTHER PUBLICATIONS

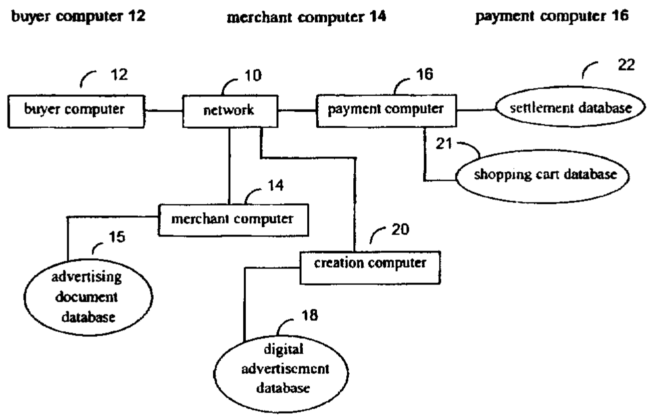
Contents of "Welcome first-time visitors" at www.amazon-.com on the Internet as of Jun. 29, 1998.

Primary Examiner—Bernarr E. Gregory
Attorney, Agent, or Firm—Fish & Richardson P.C.

[57] **ABSTRACT**

A network-based sales system includes at least one buyer computer for operation by a user desiring to buy a product, at least one merchant computer, and at least one payment computer. The buyer computer, the merchant computer, and the payment computer are interconnected by a computer network. The buyer computer is programmed to receive a user request for purchasing a product, and to cause a payment message to be sent to the payment computer that comprises a product identifier identifying the product. The payment computer is programmed to receive the payment message, to cause an access message to be created that comprises the product identifier and an access message authenticator based on a cryptographic key, and to cause the access message to be sent to the merchant computer. The merchant computer is programmed to receive the access message, to verify the access message authenticator to ensure that the access message authenticator was created using the cryptographic key, and to cause the product to be sent to the user desiring to buy the product.

38 Claims, 25 Drawing Sheets



U.S. PATENT DOCUMENTS

4,996,711	2/1991	Chaum	380/30
5,025,373	6/1991	Keyser, Jr. et al. .	
5,060,153	10/1991	Nakagawa .	
5,077,607	12/1991	Johnson et al. .	
5,105,184	4/1992	Pirani et al. .	
5,220,501	6/1993	Lawlor et al. .	
5,247,575	9/1993	Sprague et al.	380/9
5,276,736	1/1994	Chaum	380/24
5,305,195	4/1994	Murphy .	
5,311,594	5/1994	Penzias	380/24
5,319,542	6/1994	King, Jr. et al.	705/27
5,321,751	6/1994	Ray et al.	380/24
5,336,870	8/1994	Hughes	235/379
5,341,429	8/1994	Stringer et al.	380/23
5,347,632	9/1994	Filepp et al. .	
5,351,186	9/1994	Bullock et al. .	
5,351,293	9/1994	Michener et al.	380/21
5,383,113	1/1995	Kight et al. .	
5,414,833	5/1995	Hershey et al. .	
5,521,631	5/1996	Budow et al. .	
5,535,229	7/1996	Hain, Jr. et al. .	
5,557,516	9/1996	Hogan	364/406
5,557,518	9/1996	Rosen	380/24
5,557,798	9/1996	Skeen et al. .	
5,590,197	12/1996	Chen et al.	380/24
5,592,378	1/1997	Cameron et al.	705/27
5,594,910	1/1997	Filepp et al. .	
5,596,642	1/1997	Davis et al.	380/24
5,596,643	1/1997	Davis et al.	380/24
5,604,802	2/1997	Holloway	380/24
5,621,797	4/1997	Rosen	380/24
5,623,547	4/1997	Jones et al.	380/24
5,642,419	6/1997	Rosen	380/24
5,694,551	12/1997	Doyle et al.	705/26
5,715,314	2/1998	Payne et al.	380/24
5,724,424	3/1998	Gifford	380/24

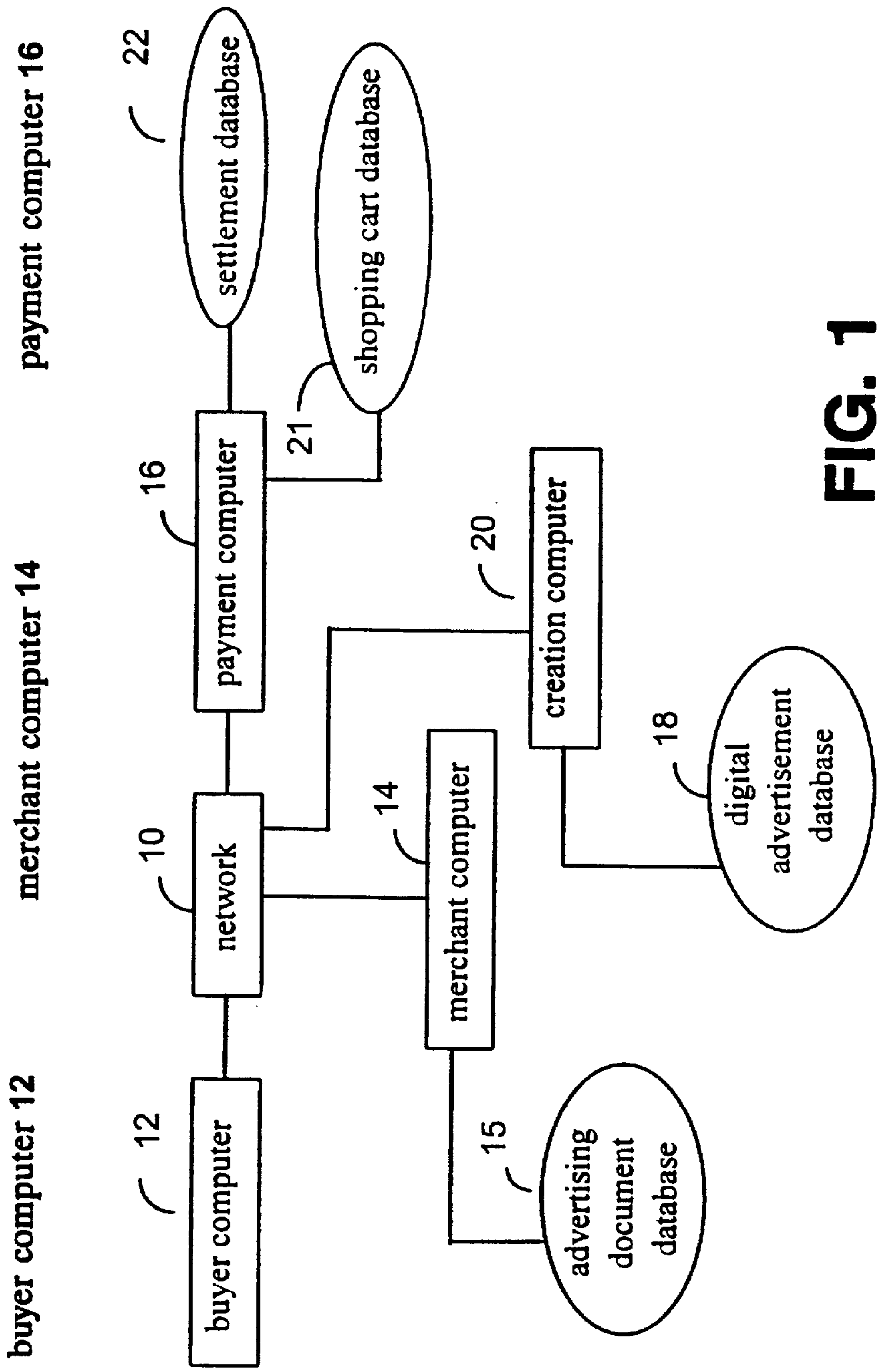

buyer computer 12
merchant computer 14
payment computer 16
buyer computer
12
network
10
payment computer
16
settlement database
22
shopping cart database
21
merchant computer
14
advertising document database
15
creation computer
20
digital advertisement database
18

FIG. 1

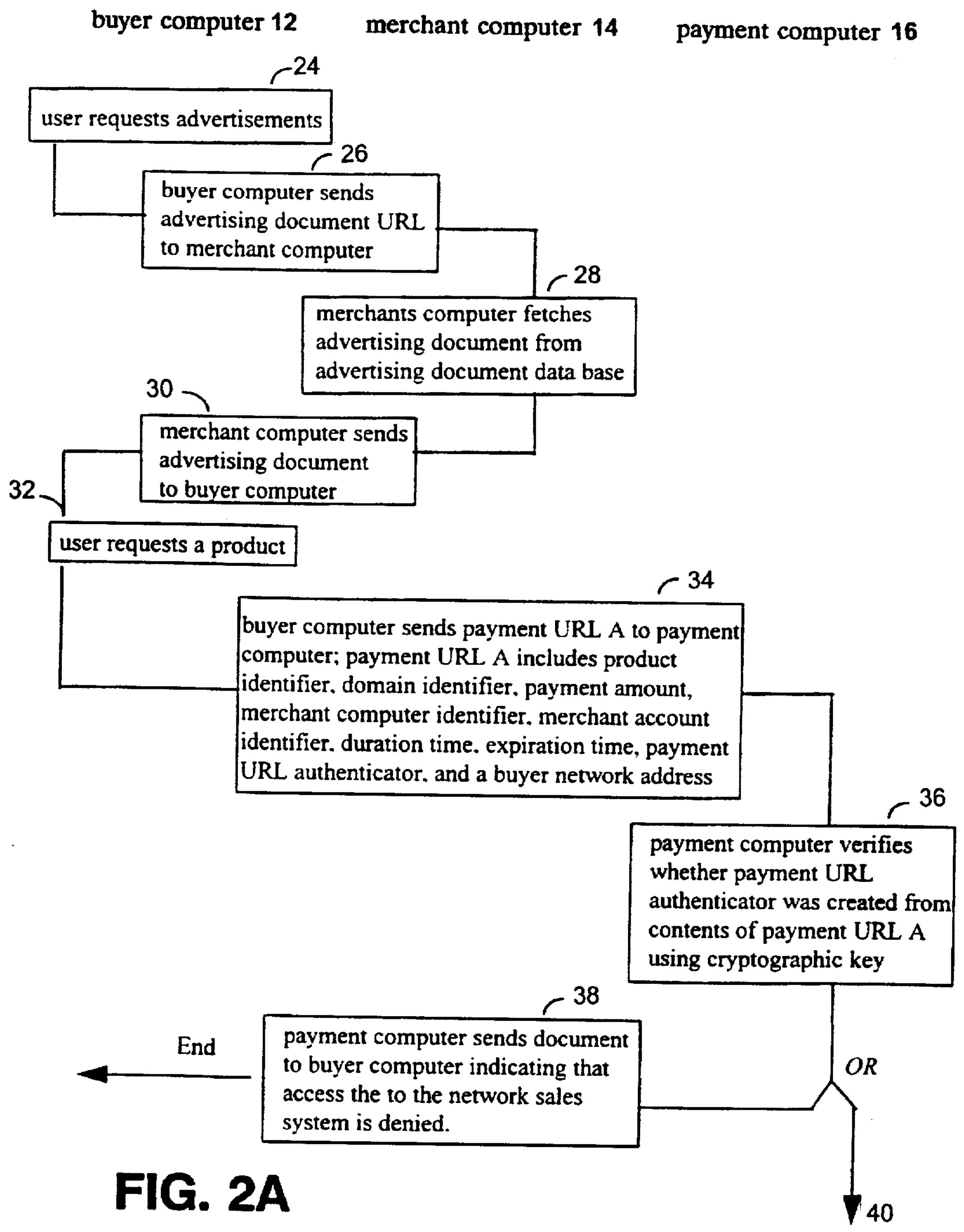

buyer computer 12
merchant computer 14
payment computer 16
24
user requests advertisements
26
buyer computer sends advertising document URL to merchant computer
28
merchants computer fetches advertising document from advertising document data base
30
merchant computer sends advertising document to buyer computer
32
user requests a product
34
buyer computer sends payment URL A to payment computer; payment URL A includes product identifier, domain identifier, payment amount, merchant computer identifier, merchant account identifier, duration time, expiration time, payment URL authenticator, and a buyer network address
36
payment computer verifies whether payment URL authenticator was created from contents of payment URL A using cryptographic key
38
payment computer sends document to buyer computer indicating that access the to the network sales system is denied.
End
OR
40

FIG. 2A

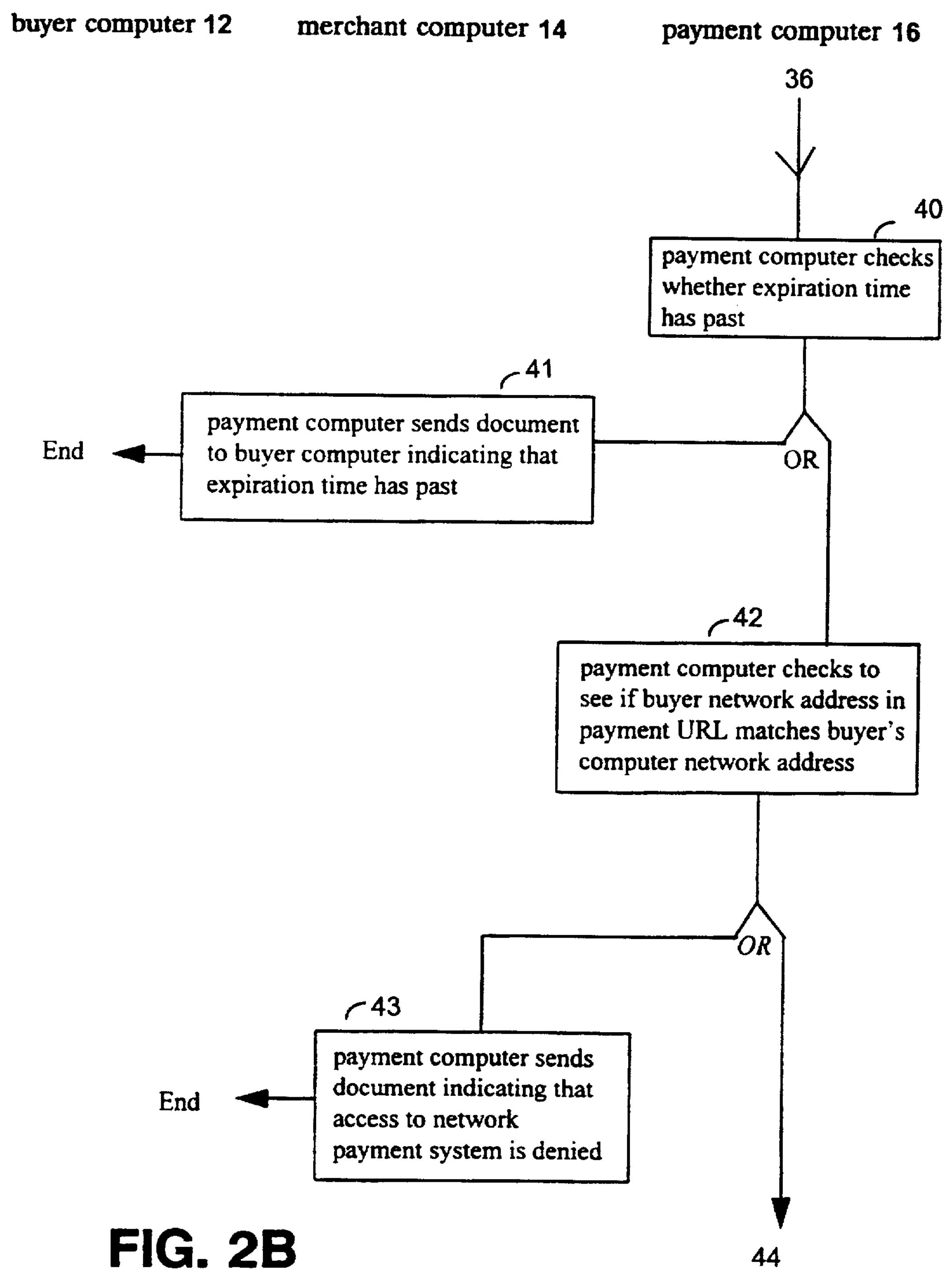

buyer computer 12
merchant computer 14
payment computer 16
36
40
payment computer checks whether expiration time has past
41
End
payment computer sends document to buyer computer indicating that expiration time has past
OR
42
payment computer checks to see if buyer network address in payment URL matches buyer's computer network address
OR
43
End
payment computer sends document indicating that access to network payment system is denied
44

FIG. 2B

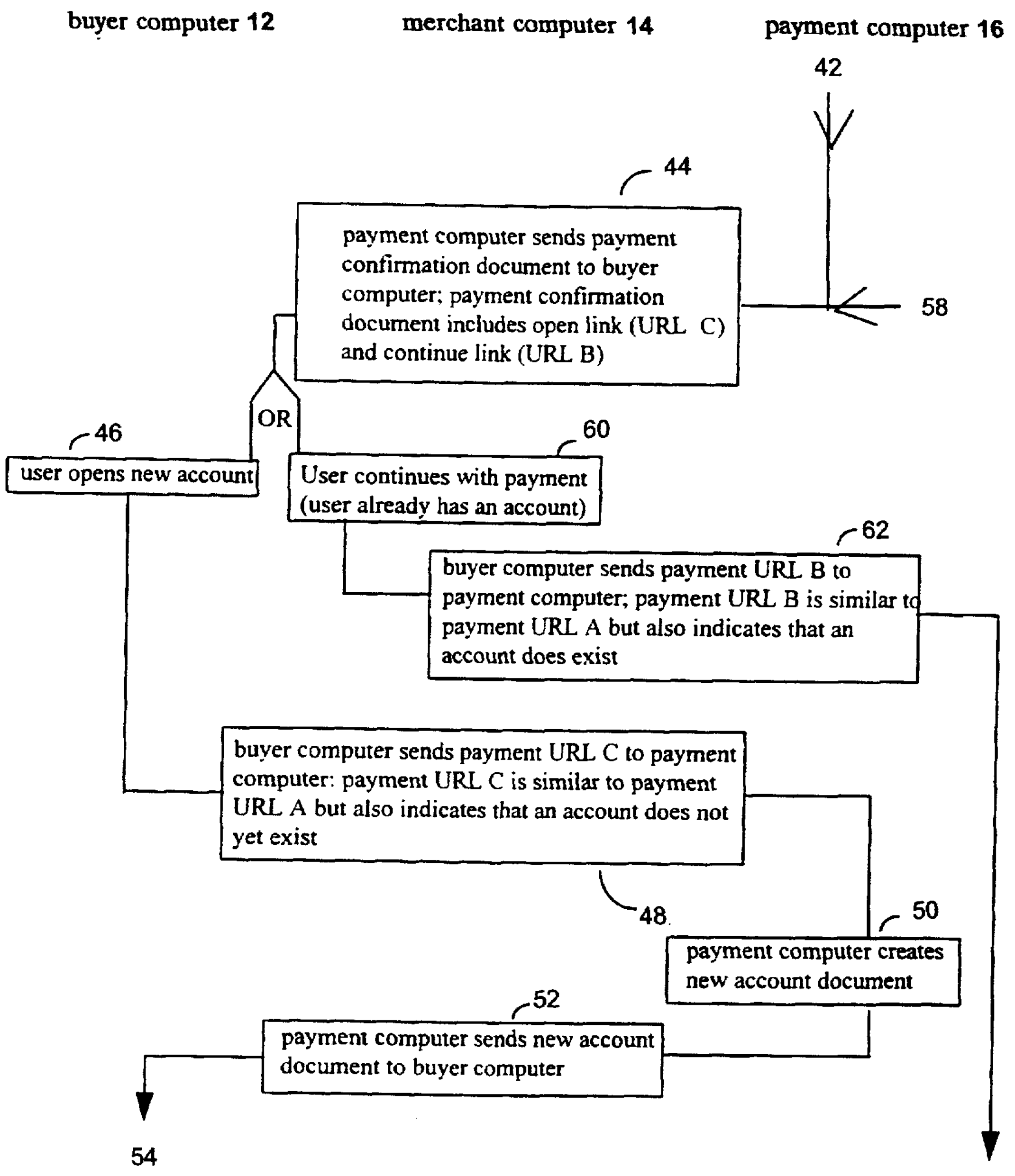

buyer computer 12
merchant computer 14
payment computer 16
42
44
payment computer sends payment confirmation document to buyer computer; payment confirmation document includes open link (URL C) and continue link (URL B)
58
OR
46
user opens new account
60
User continues with payment (user already has an account)
62
buyer computer sends payment URL B to payment computer; payment URL B is similar to payment URL A but also indicates that an account does exist
buyer computer sends payment URL C to payment computer: payment URL C is similar to payment URL A but also indicates that an account does not yet exist
48
50
payment computer creates new account document
52
payment computer sends new account document to buyer computer
54
64

FIG. 2C

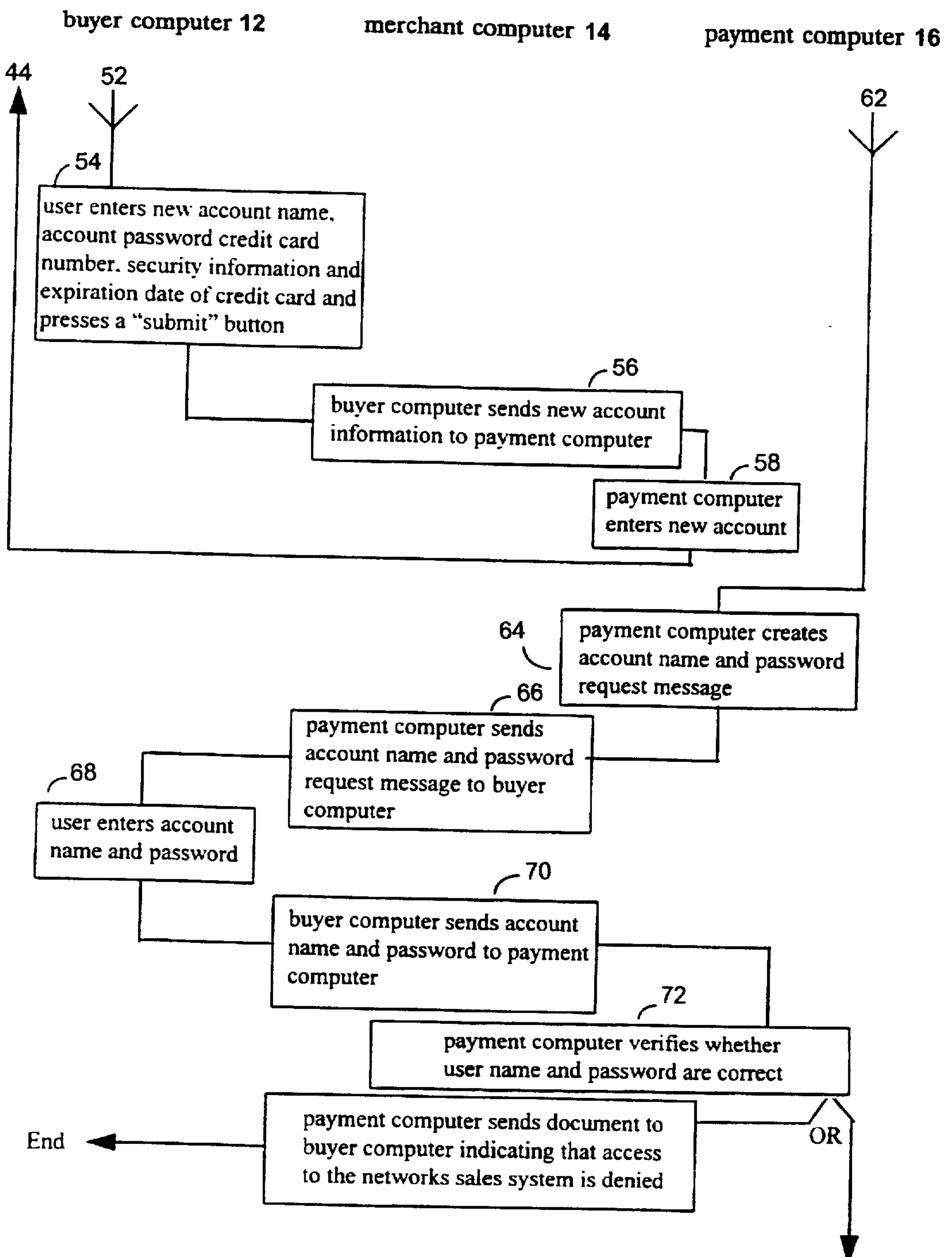

buyer computer 12
merchant computer 14
payment computer 16
44
52
62
54
user enters new account name, account password credit card number. security information and expiration date of credit card and presses a "submit" button
56
buyer computer sends new account information to payment computer
58
payment computer enters new account
64
payment computer creates account name and password request message
66
payment computer sends account name and password request message to buyer computer
68
user enters account name and password
70
buyer computer sends account name and password to payment computer
72
payment computer verifies whether user name and password are correct
payment computer sends document to buyer computer indicating that access to the networks sales system is denied
End
OR
73

FIG. 2D

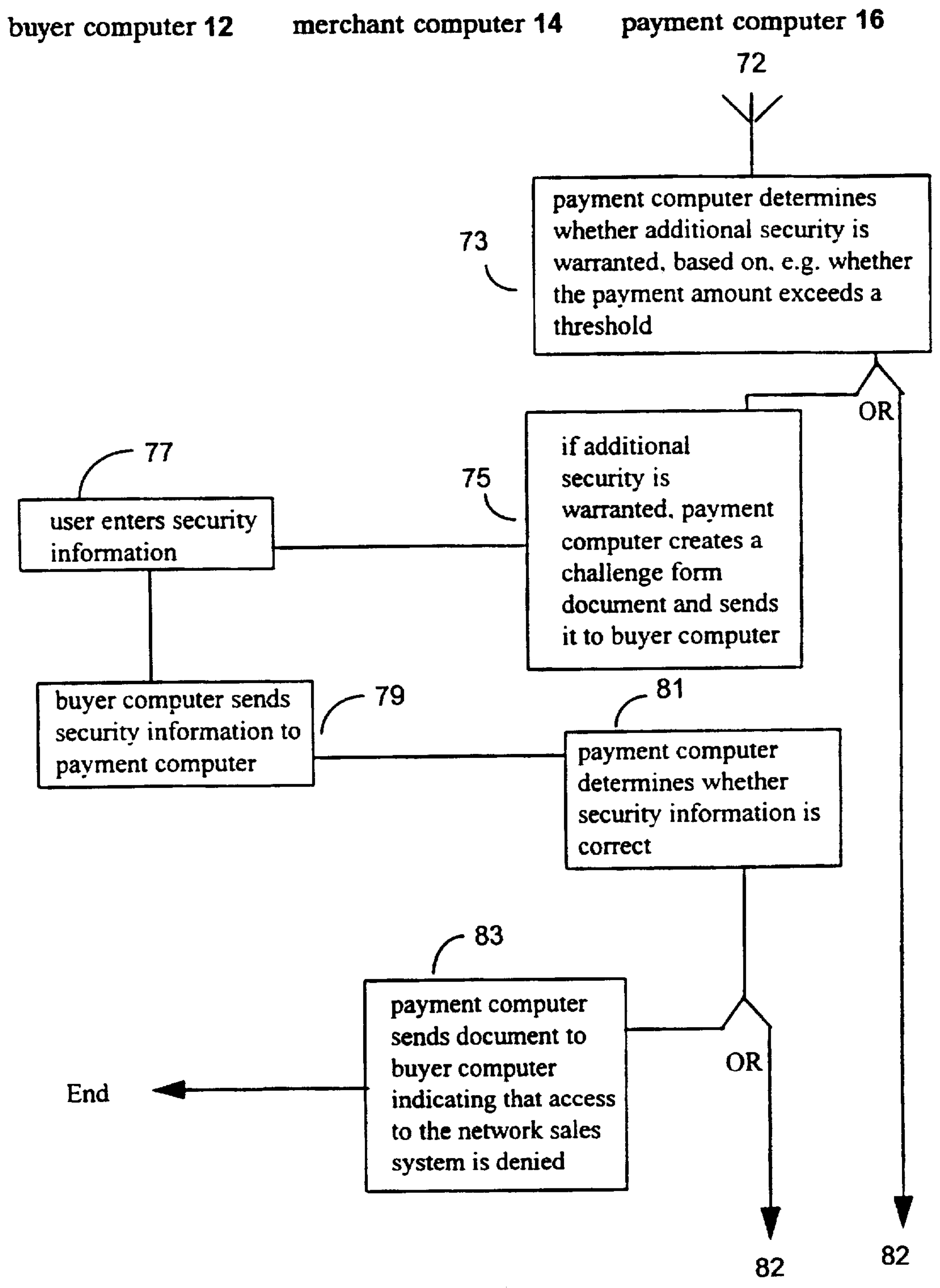

buyer computer 12
merchant computer 14
payment computer 16
72
73
payment computer determines whether additional security is warranted, based on, e.g. whether the payment amount exceeds a threshold
OR
77
75
if additional security is warranted, payment computer creates a challenge form document and sends it to buyer computer
user enters security information
buyer computer sends security information to payment computer
79
81
payment computer determines whether security information is correct
83
payment computer sends document to buyer computer indicating that access to the network sales system is denied
End
OR
82
82

FIG. 2E

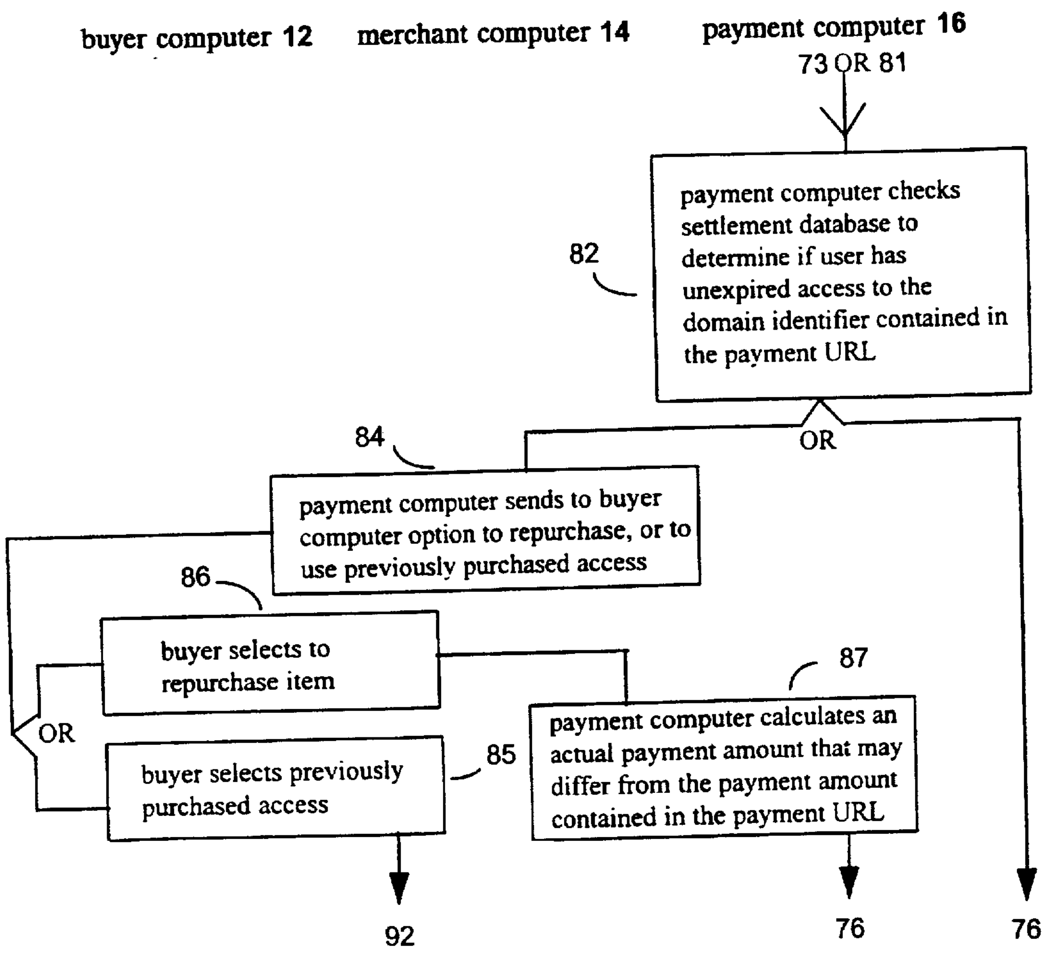

buyer computer 12
merchant computer 14
payment computer 16
73 OR 81
82
payment computer checks settlement database to determine if user has unexpired access to the domain identifier contained in the payment URL
OR
84
payment computer sends to buyer computer option to repurchase, or to use previously purchased access
86
buyer selects to repurchase item
OR
buyer selects previously purchased access
85
87
payment computer calculates an actual payment amount that may differ from the payment amount contained in the payment URL
92
76
76

FIG. 2F

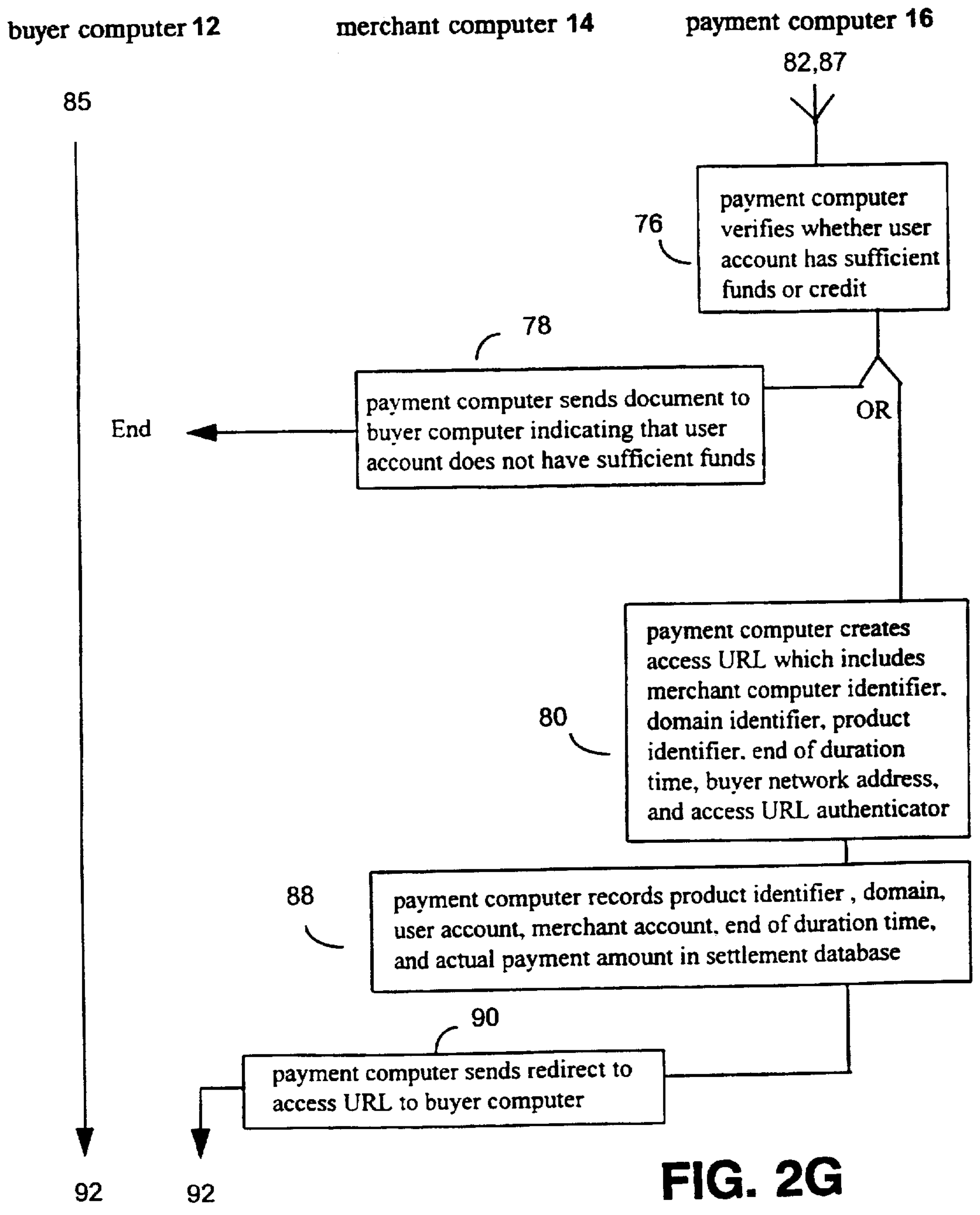

buyer computer 12
merchant computer 14
payment computer 16
82,87
85
76
payment computer verifies whether user account has sufficient funds or credit
78
payment computer sends document to buyer computer indicating that user account does not have sufficient funds
End
OR
80
payment computer creates access URL which includes merchant computer identifier, domain identifier, product identifier, end of duration time, buyer network address, and access URL authenticator
88
payment computer records product identifier , domain, user account, merchant account, end of duration time, and actual payment amount in settlement database
90
payment computer sends redirect to access URL to buyer computer
92
92

FIG. 2G

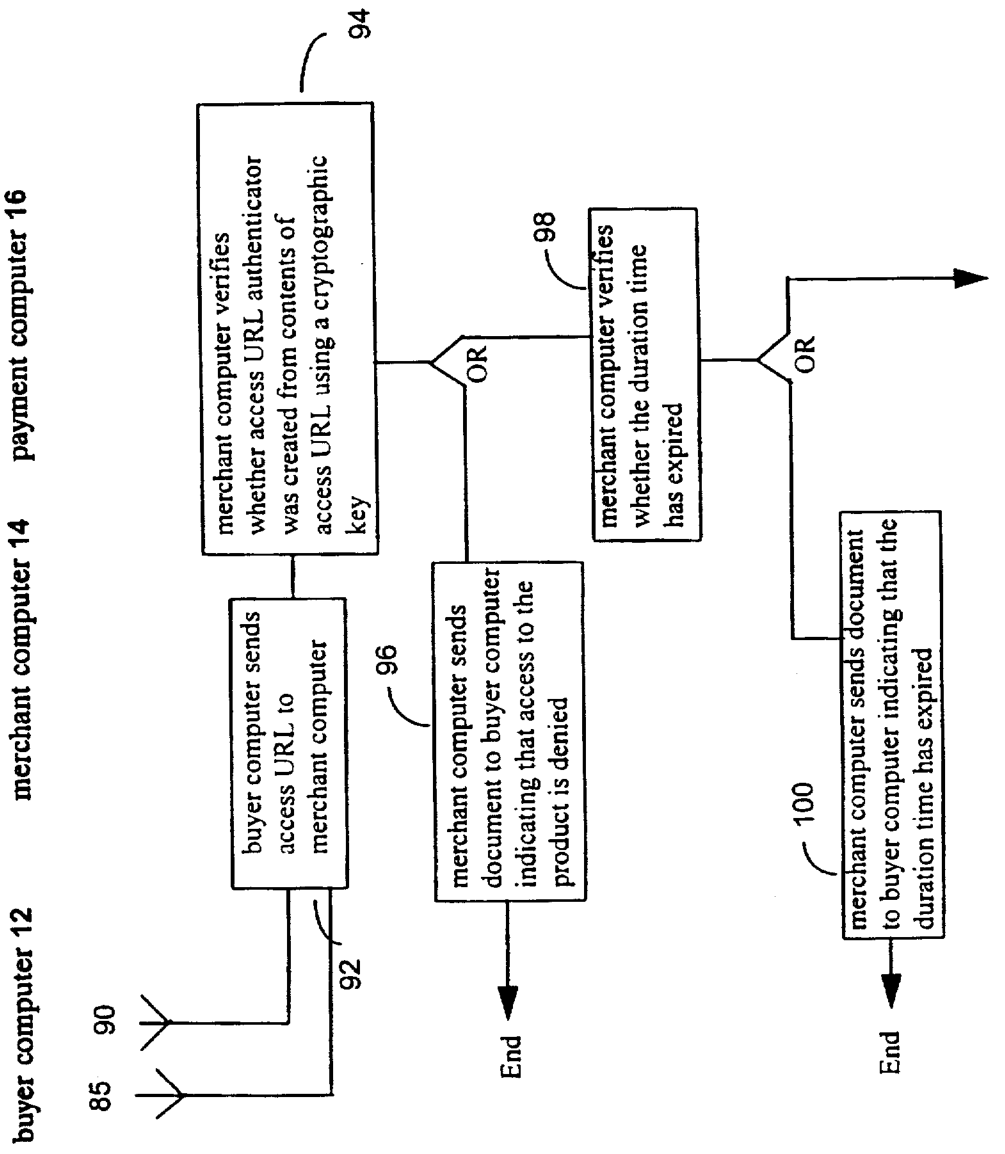

buyer computer 12
merchant computer 14
payment computer 16
85
90
92
buyer computer sends access URL to merchant computer
94
merchant computer verifies whether access URL authenticator was created from contents of access URL using a cryptographic key
OR
96
merchant computer sends document to buyer computer indicating that access to the product is denied
End
98
merchant computer verifies whether the duration time has expired
OR
101
100
merchant computer sends document to buyer computer indicating that the duration time has expired
End

FIG. 2H

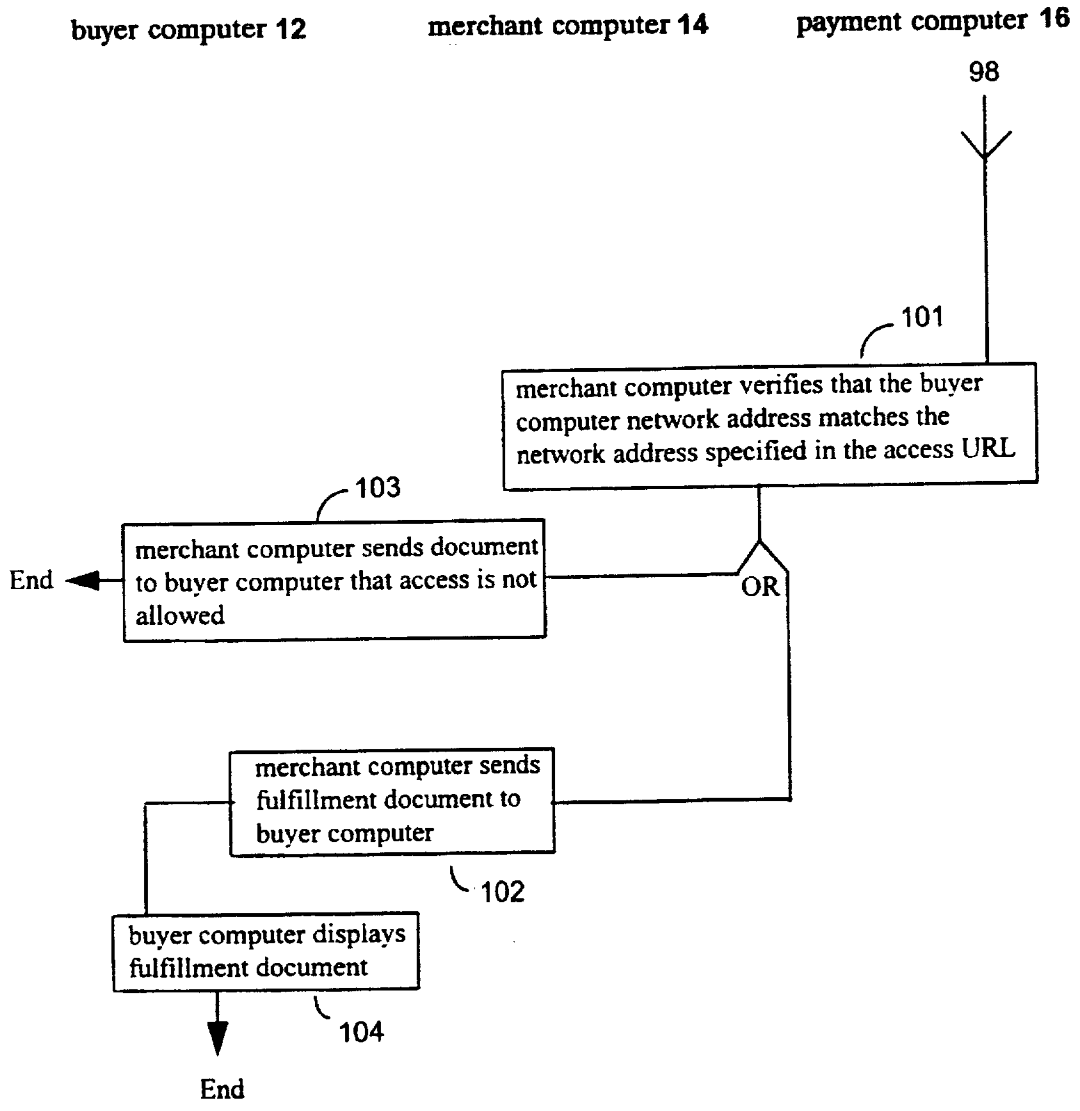

buyer computer 12
merchant computer 14
payment computer 16
98
101
merchant computer verifies that the buyer computer network address matches the network address specified in the access URL
103
merchant computer sends document to buyer computer that access is not allowed
End
OR
merchant computer sends fulfillment document to buyer computer
102
buyer computer displays fulfillment document
104
End

FIG. 2I

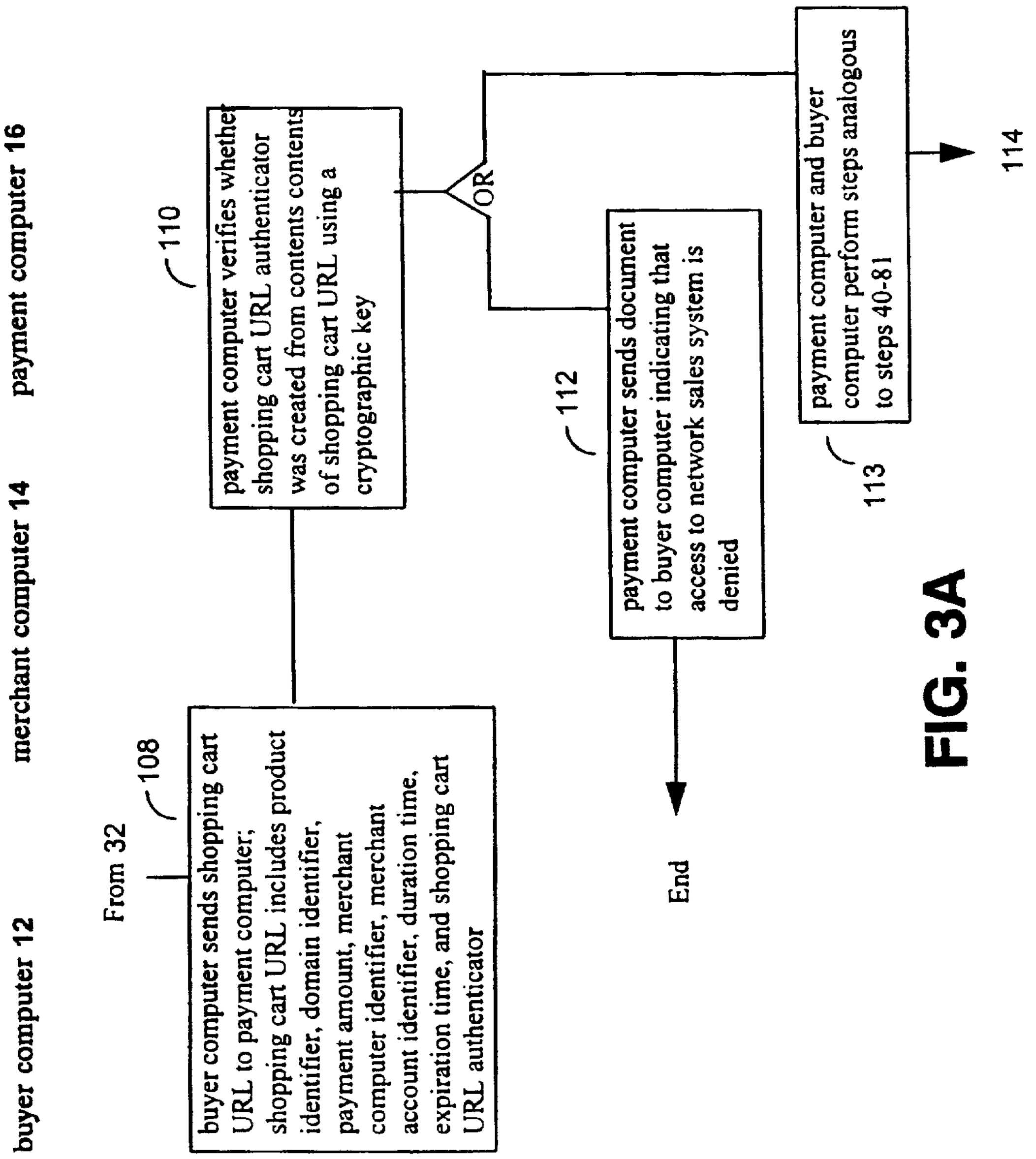

buyer computer 12
merchant computer 14
payment computer 16
From 32
108
buyer computer sends shopping cart URL to payment computer; shopping cart URL includes product identifier, domain identifier, payment amount, merchant computer identifier, merchant account identifier, duration time, expiration time, and shopping cart URL authenticator
110
payment computer verifies whether shopping cart URL authenticator was created from contents contents of shopping cart URL using a cryptographic key
OR
112
payment computer sends document to buyer computer indicating that access to network sales system is denied
End
113
payment computer and buyer computer perform steps analogous to steps 40-81
114

FIG. 3A

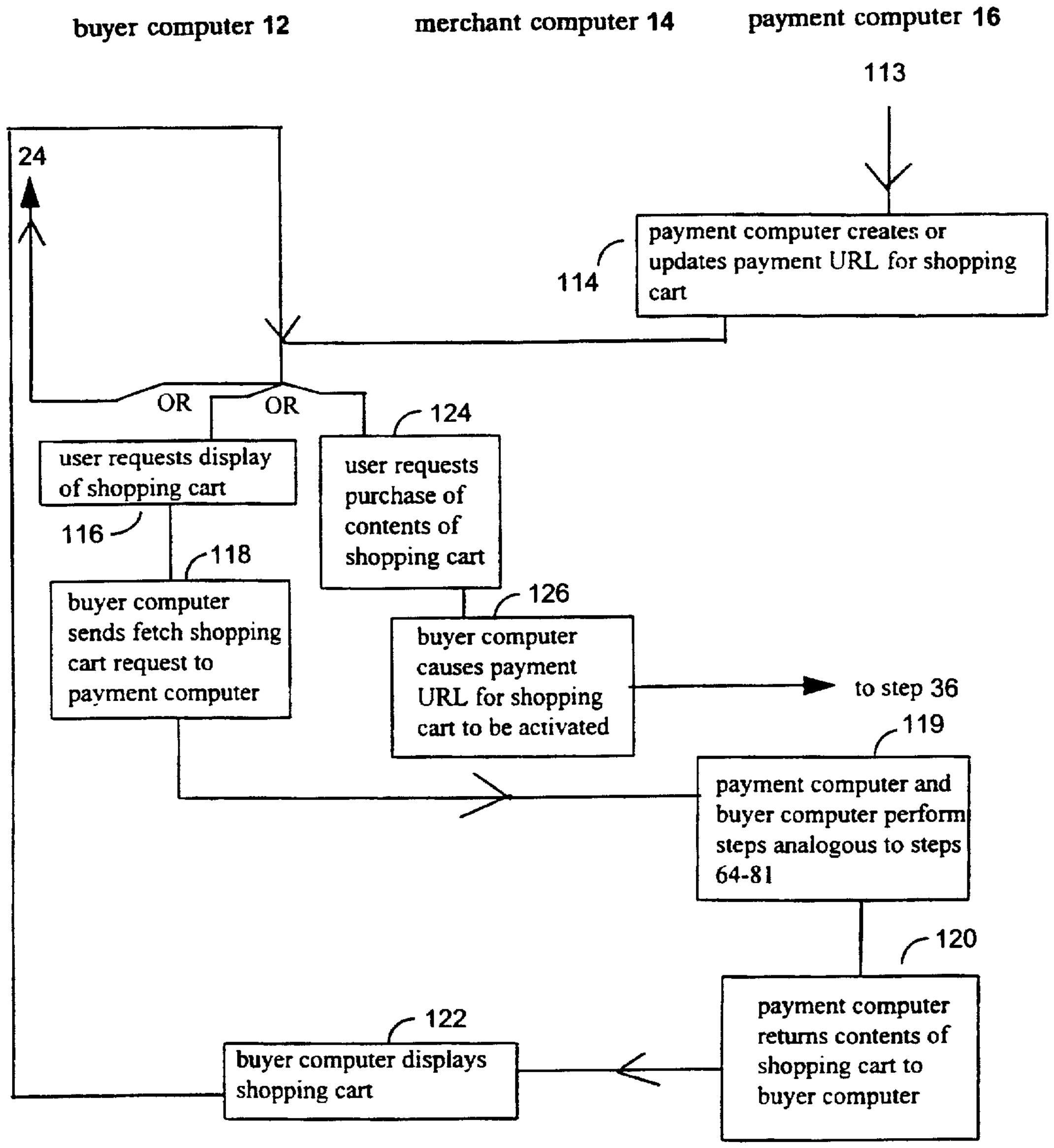

buyer computer 12
merchant computer 14
payment computer 16
113
24
payment computer creates or updates payment URL for shopping cart
114
OR
OR
124
user requests display of shopping cart
116
user requests purchase of contents of shopping cart
118
buyer computer sends fetch shopping cart request to payment computer
126
buyer computer causes payment URL for shopping cart to be activated
to step 36
119
payment computer and buyer computer perform steps analogous to steps 64-81
120
122
payment computer returns contents of shopping cart to buyer computer
buyer computer displays shopping cart

FIG. 3B

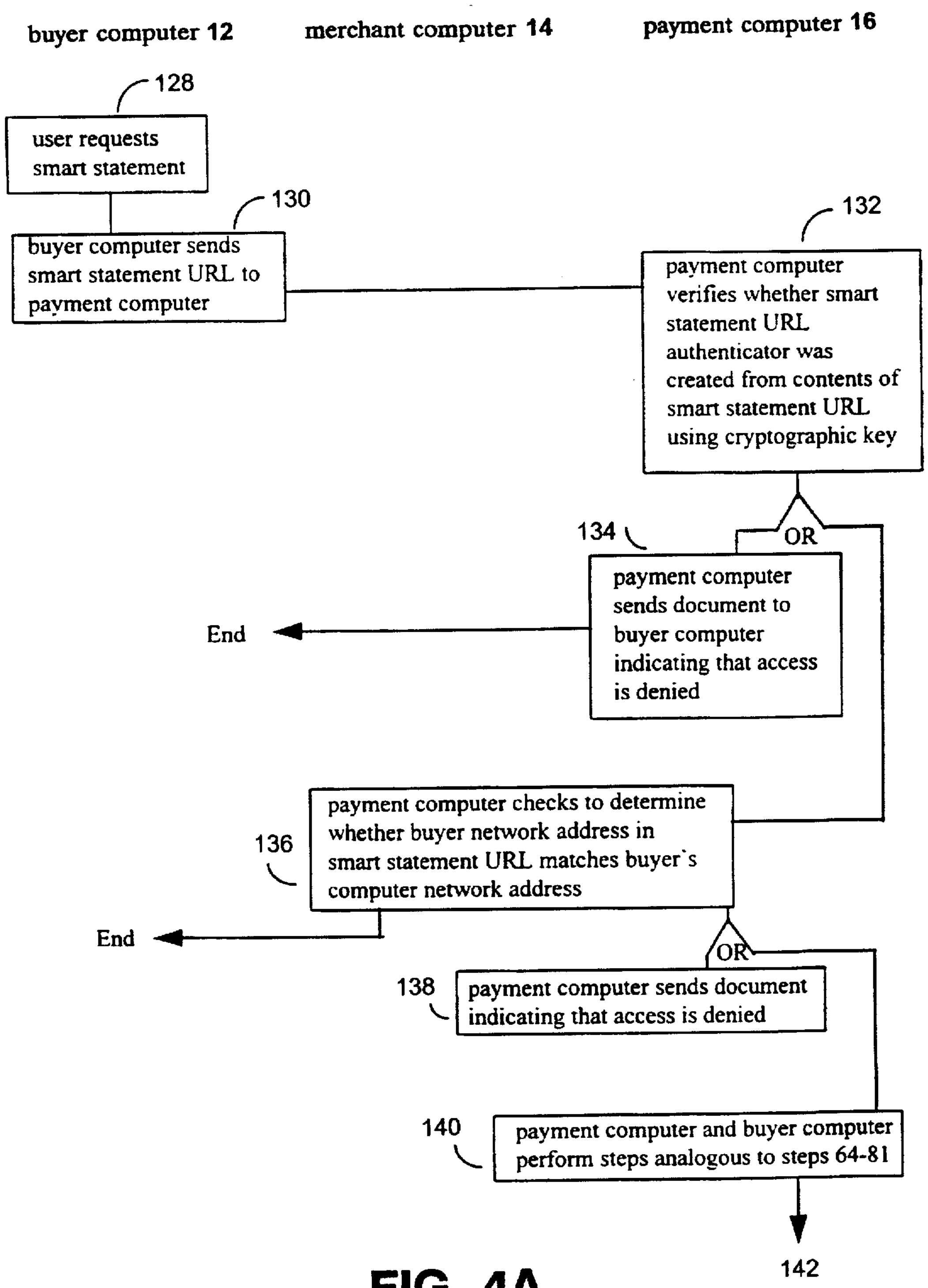

buyer computer 12
merchant computer 14
payment computer 16
128
user requests smart statement
130
buyer computer sends smart statement URL to payment computer
132
payment computer verifies whether smart statement URL authenticator was created from contents of smart statement URL using cryptographic key
OR
134
payment computer sends document to buyer computer indicating that access is denied
End
136
payment computer checks to determine whether buyer network address in smart statement URL matches buyer's computer network address
End
OR
138
payment computer sends document indicating that access is denied
140
payment computer and buyer computer perform steps analogous to steps 64-81
142

FIG. 4A

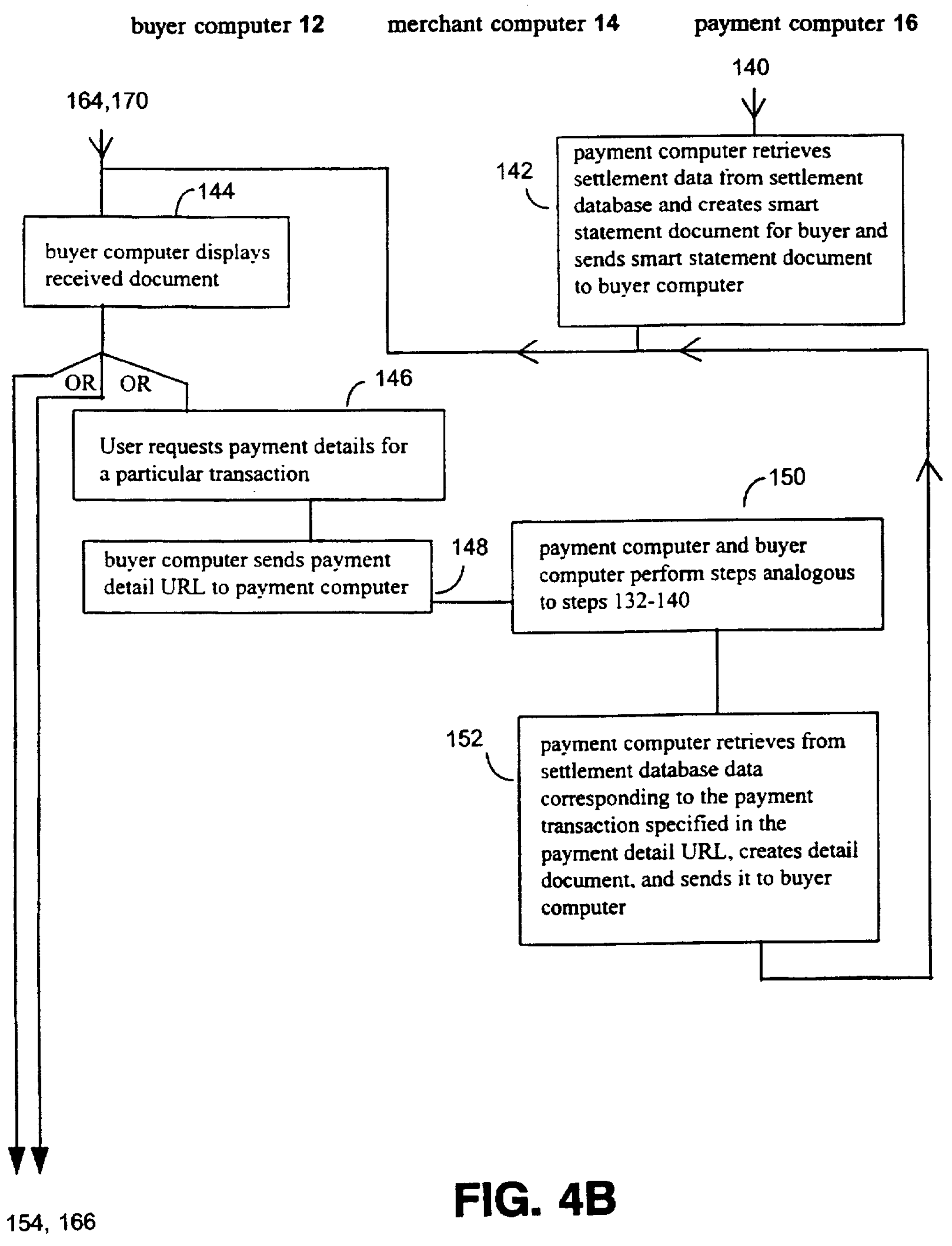

buyer computer 12
merchant computer 14
payment computer 16
140
164,170
142
payment computer retrieves settlement data from settlement database and creates smart statement document for buyer and sends smart statement document to buyer computer
144
buyer computer displays received document
OR
OR
146
User requests payment details for a particular transaction
150
148
buyer computer sends payment detail URL to payment computer
payment computer and buyer computer perform steps analogous to steps 132-140
152
payment computer retrieves from settlement database data corresponding to the payment transaction specified in the payment detail URL, creates detail document, and sends it to buyer computer
154, 166

FIG. 4B

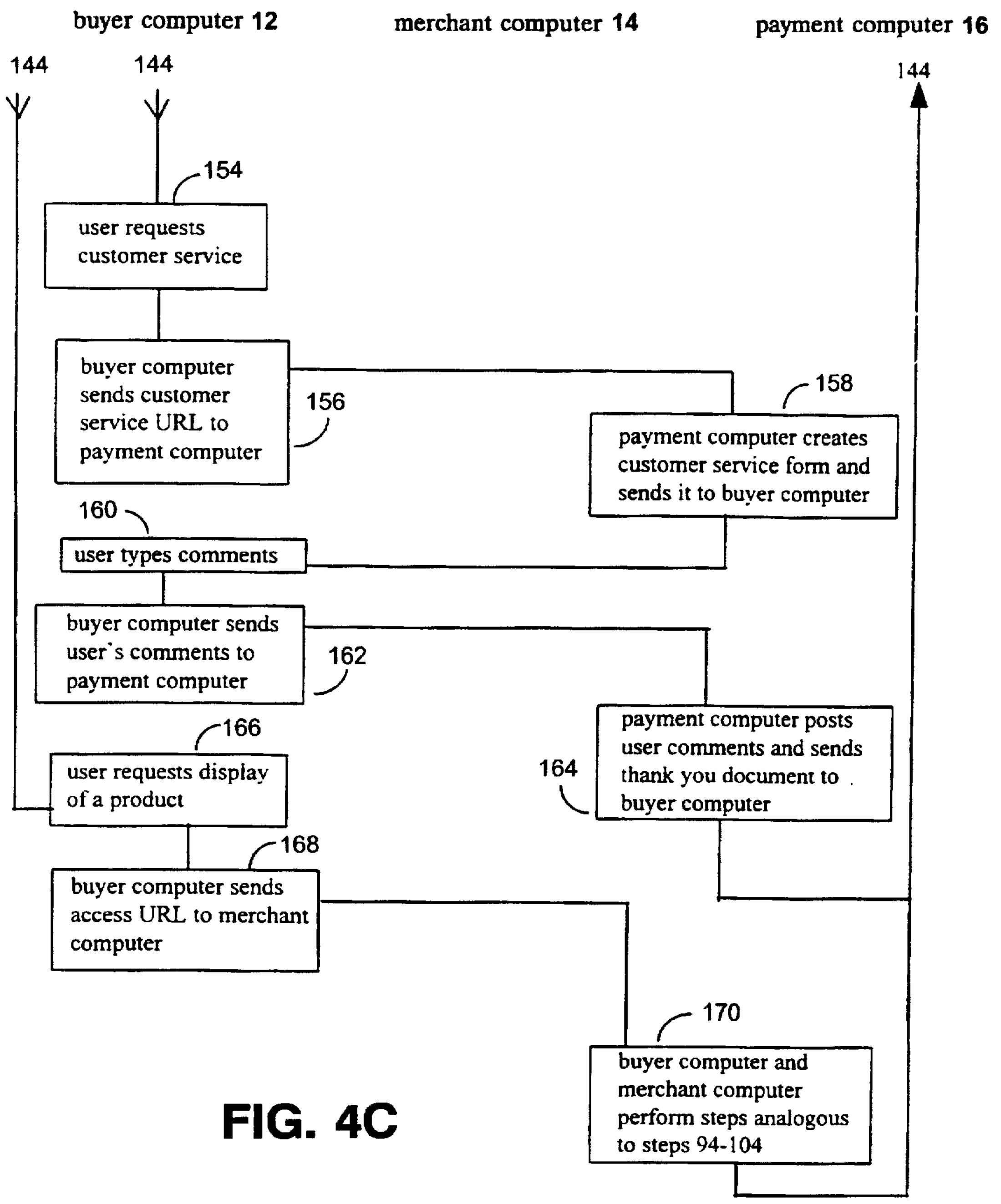
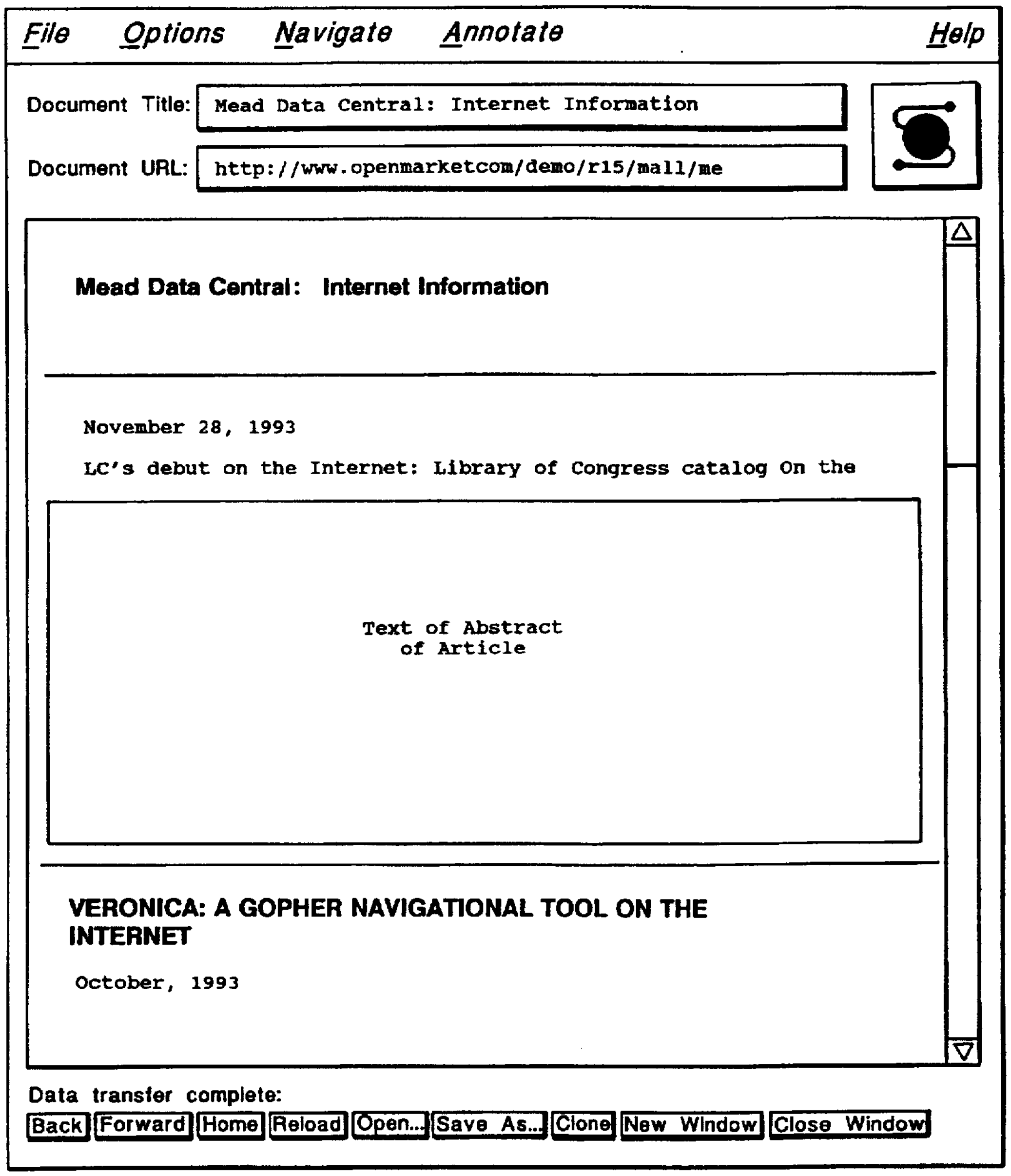

buyer computer 12
merchant computer 14
payment computer 16
144
144
144
154
user requests customer service
buyer computer sends customer service URL to payment computer
156
158
payment computer creates customer service form and sends it to buyer computer
160
user types comments
buyer computer sends user's comments to payment computer
162
payment computer posts user comments and sends thank you document to buyer computer
164
166
user requests display of a product
168
buyer computer sends access URL to merchant computer
170
buyer computer and merchant computer perform steps analogous to steps 94-104

FIG. 4C



Mead Data Central: Internet Information

http://www.openmarketcom/demo/r15/mall/me

Mead Data Central: Internet Information

November 28, 1993

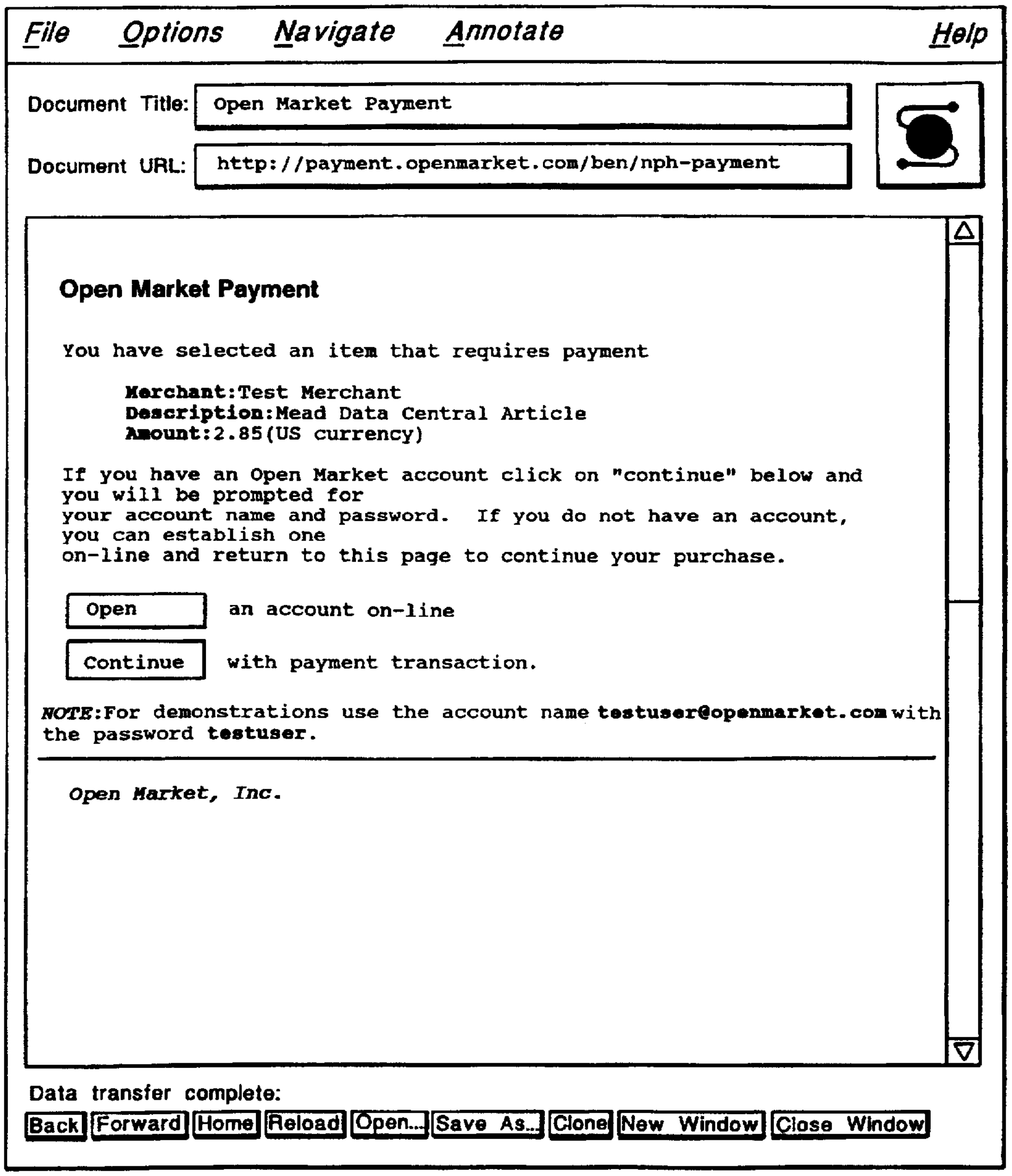
LC's debut on the Internet: Library of Congress catalog On the

Text of Abstract
of Article

VERONICA: A GOPHER NAVIGATIONAL TOOL ON THE INTERNET

October, 1993

FIG. 5



Document Title: Open Market Payment

Document URL: http://payment.openmarket.com/ben/nph-payment

Open Market Payment

You have selected an item that requires payment

Merchant: Test Merchant
Description: Mead Data Central Article
Amount:2.85(US currency)

If you have an Open Market account click on "continue" below and you will be prompted for your account name and password. If you do not have an account, you can establish one on-line and return to this page to continue your purchase.

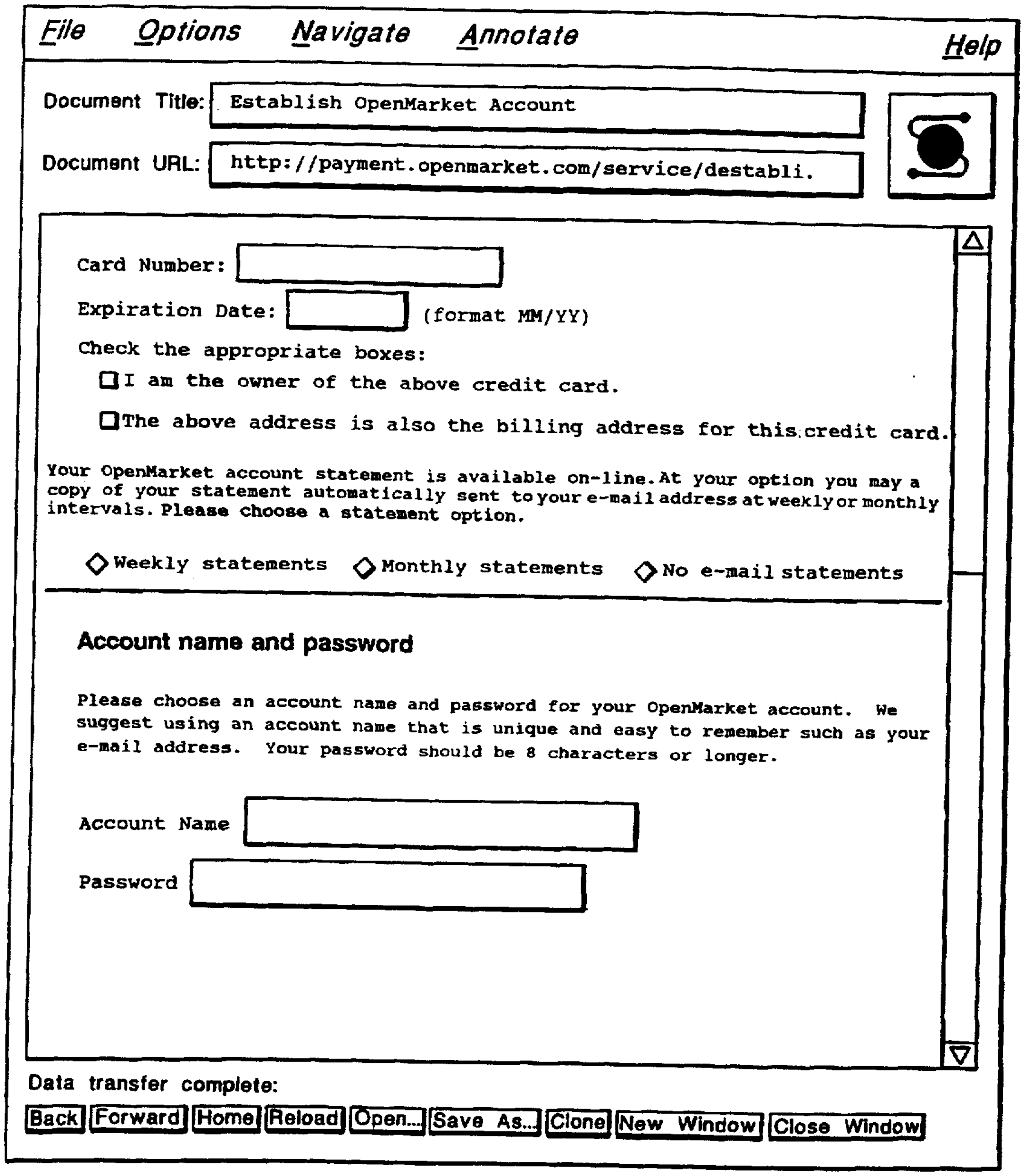
Open an account on-line

Continue with payment transaction.

NOTE:For demonstrations use the account name **testuser@openmarket.com** with the password **testuser**.

Open Market, Inc.

FIG. 6



Document Title: Establish OpenMarket Account

Document URL: http://payment.openmarket.com/service/destabli.

Card Number:

Expiration Date: (format MM/YY)

Check the appropriate boxes:

☐ I am the owner of the above credit card.

☐ The above address is also the billing address for this credit card.

Your OpenMarket account statement is available on-line. At your option you may a copy of your statement automatically sent to your e-mail address at weekly or monthly intervals. Please choose a statement option.

◇ Weekly statements ◇ Monthly statements ◇ No e-mail statements

Account name and password

Please choose an account name and password for your OpenMarket account. We suggest using an account name that is unique and easy to remember such as your e-mail address. Your password should be 8 characters or longer.

Account Name

Password

Data transfer complete:

FIG. 7

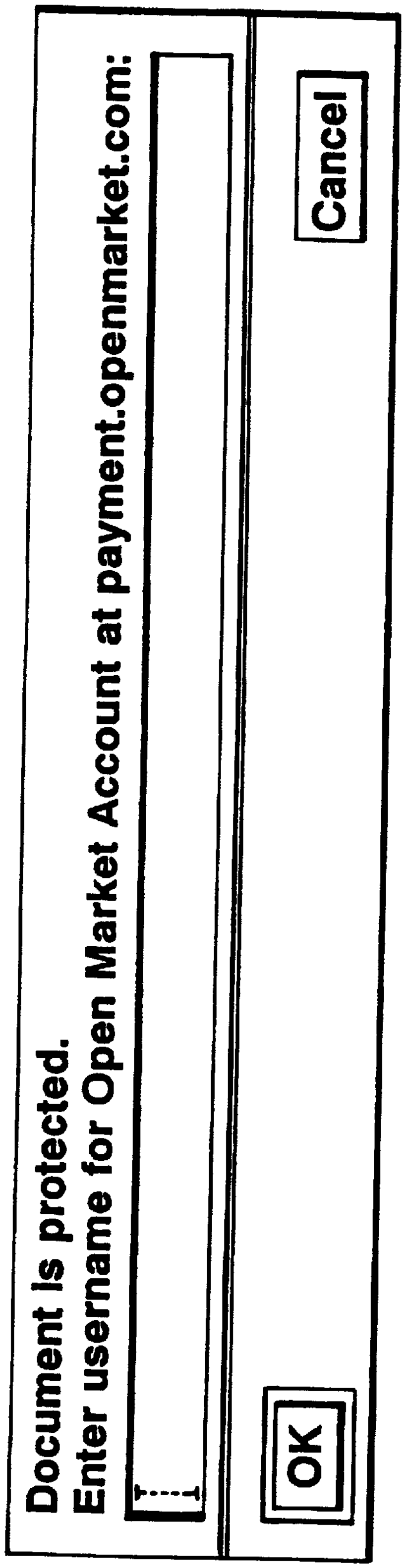
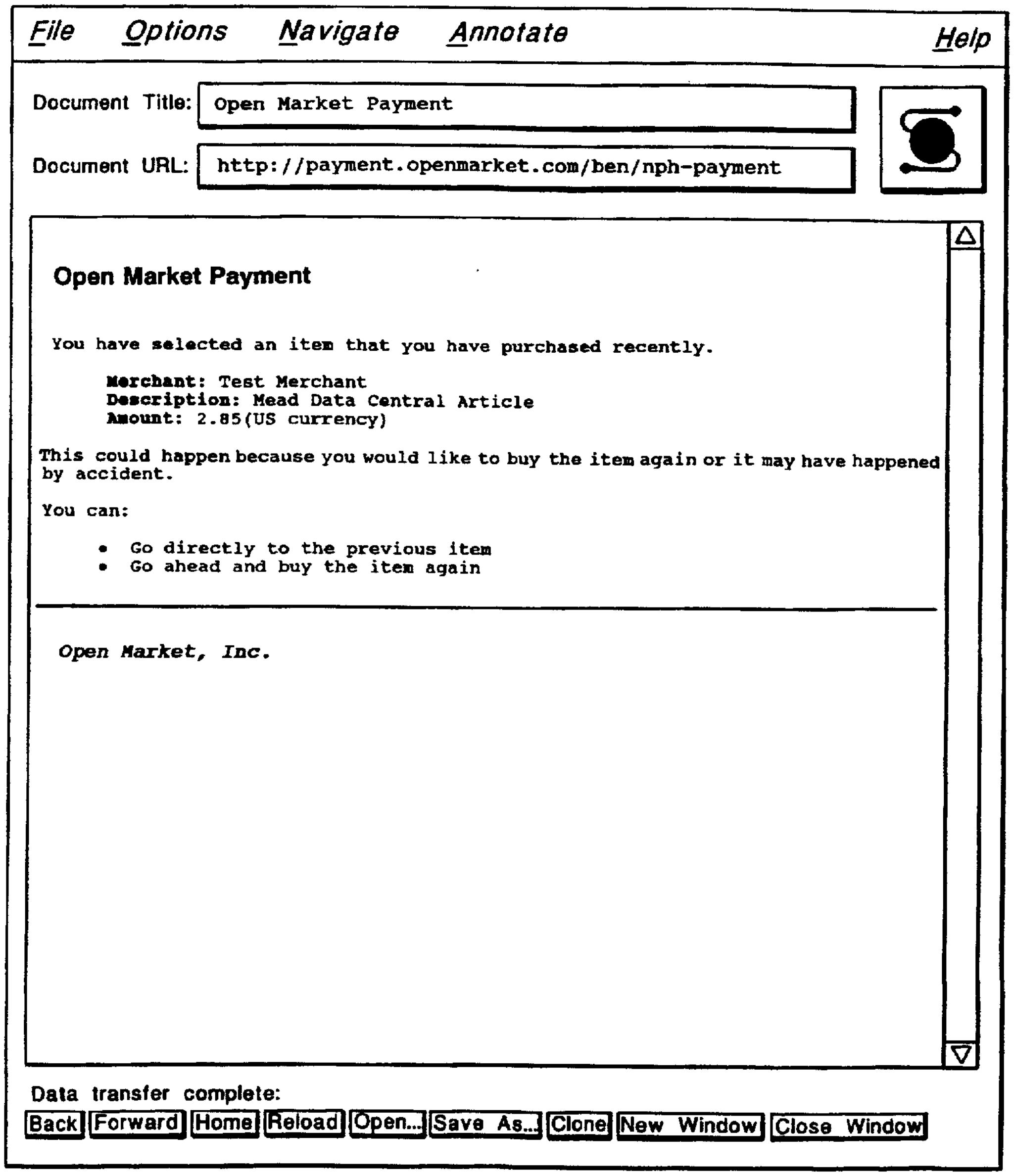

Document Is protected.
Enter username for Open Market Account at payment.openmarket.com:

FIG. 8



Document Title: Open Market Payment

Document URL: http://payment.openmarket.com/ben/nph-payment

Open Market Payment

You have selected an item that you have purchased recently.

Merchant: Test Merchant
Description: Mead Data Central Article
Amount: 2.85(US currency)

This could happen because you would like to buy the item again or it may have happened by accident.

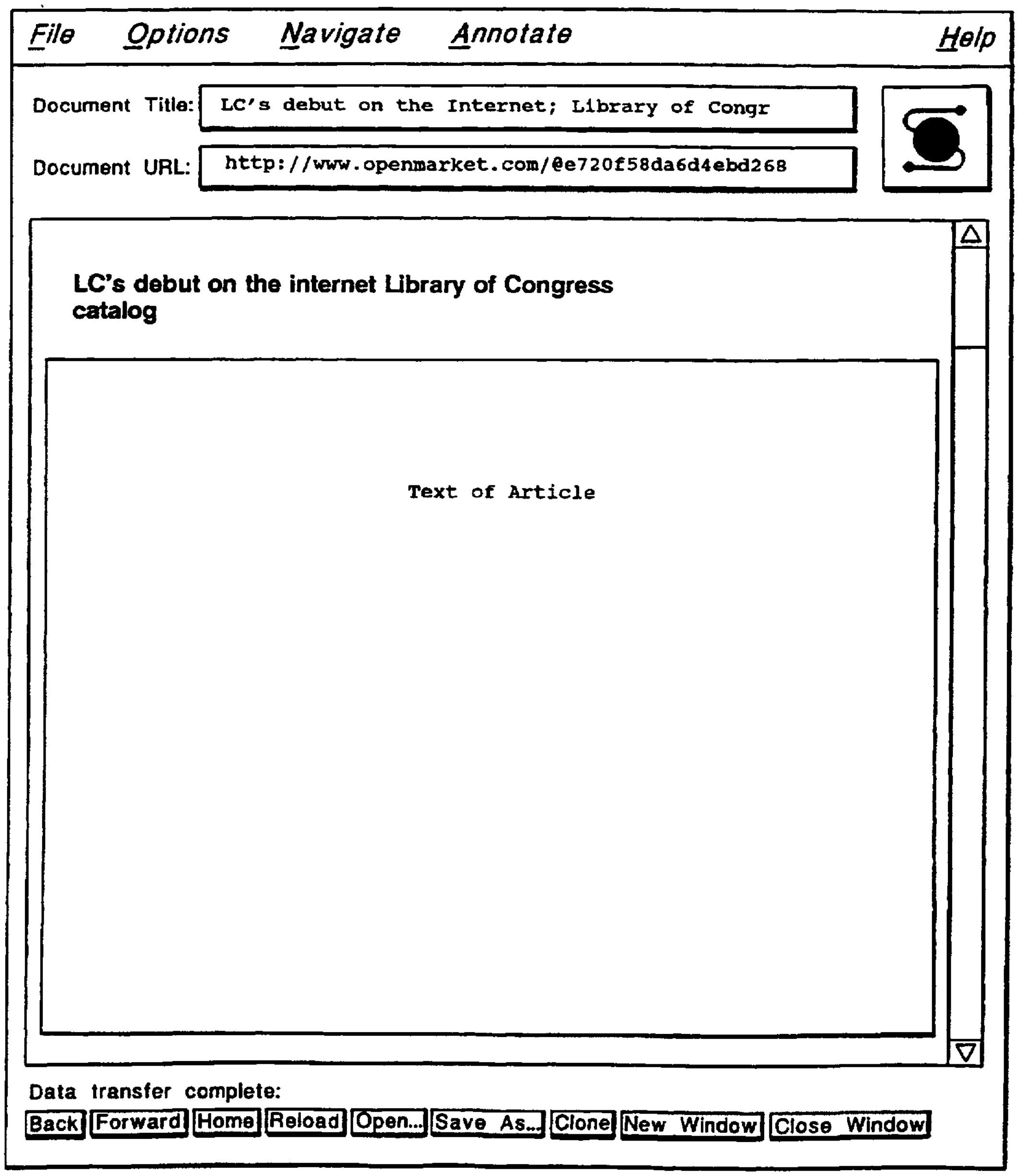
You can:

- Go directly to the previous item
- Go ahead and buy the item again

Open Market, Inc.

Data transfer complete:

FIG. 9



Document Title: LC's debut on the Internet; Library of Congr

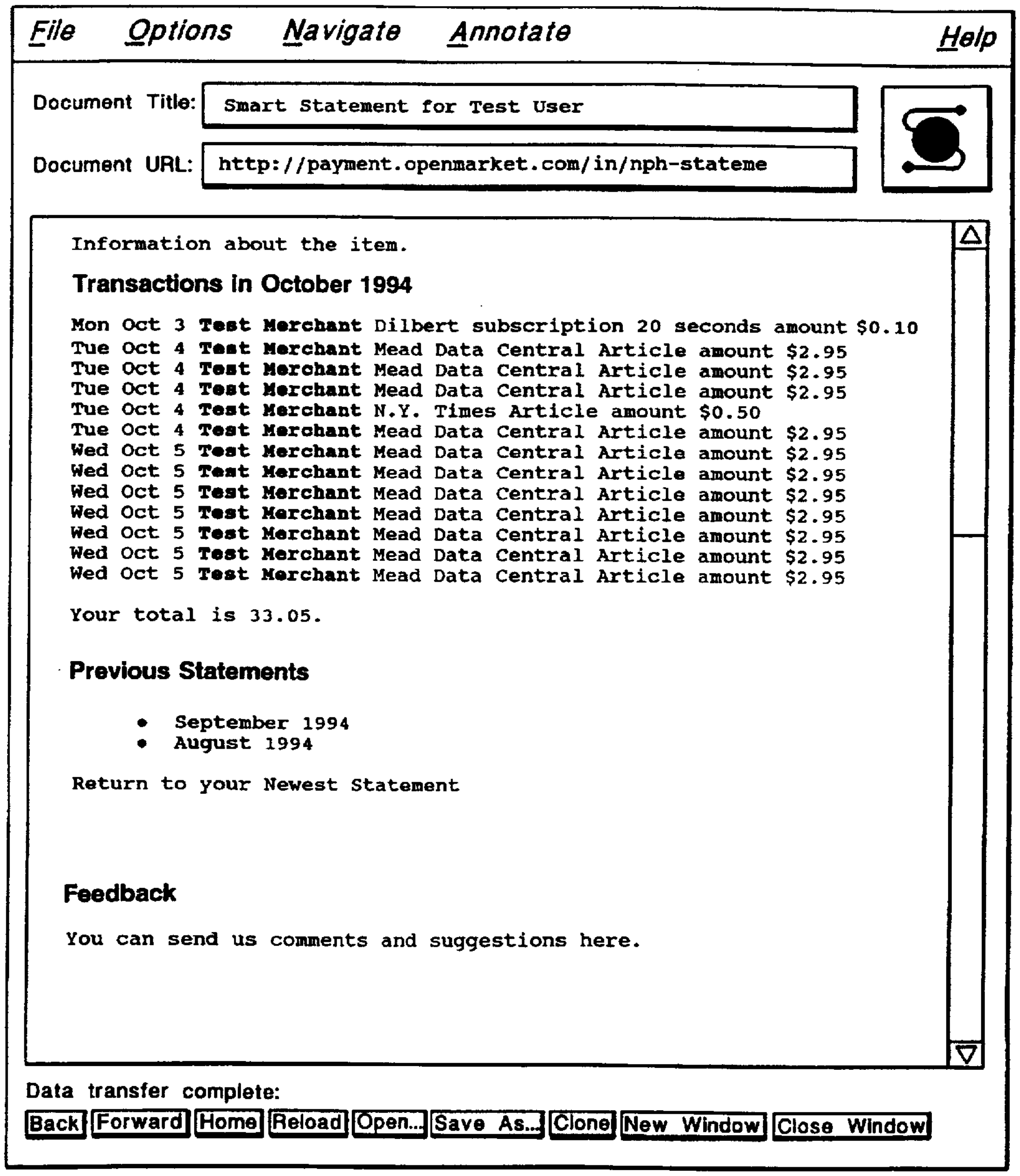
Document URL: http://www.openmarket.com/@e720f58da6d4ebd268

LC's debut on the internet Library of Congress catalog

Text of Article

Data transfer complete:

FIG. 10



Document Title: Smart Statement for Test User

Document URL: http://payment.openmarket.com/in/nph-stateme

Information about the item.

Transactions in October 1994

```
Mon Oct 3 Test Merchant Dilbert subscription 20 seconds amount $0.10
Tue Oct 4 Test Merchant Mead Data Central Article amount $2.95
Tue Oct 4 Test Merchant Mead Data Central Article amount $2.95
Tue Oct 4 Test Merchant Mead Data Central Article amount $2.95
Tue Oct 4 Test Merchant N.Y. Times Article amount $0.50
Tue Oct 4 Test Merchant Mead Data Central Article amount $2.95
Wed Oct 5 Test Merchant Mead Data Central Article amount $2.95
Wed Oct 5 Test Merchant Mead Data Central Article amount $2.95
Wed Oct 5 Test Merchant Mead Data Central Article amount $2.95
Wed Oct 5 Test Merchant Mead Data Central Article amount $2.95
Wed Oct 5 Test Merchant Mead Data Central Article amount $2.95
Wed Oct 5 Test Merchant Mead Data Central Article amount $2.95
Wed Oct 5 Test Merchant Mead Data Central Article amount $2.95
```

Your total is 33.05.

Previous Statements

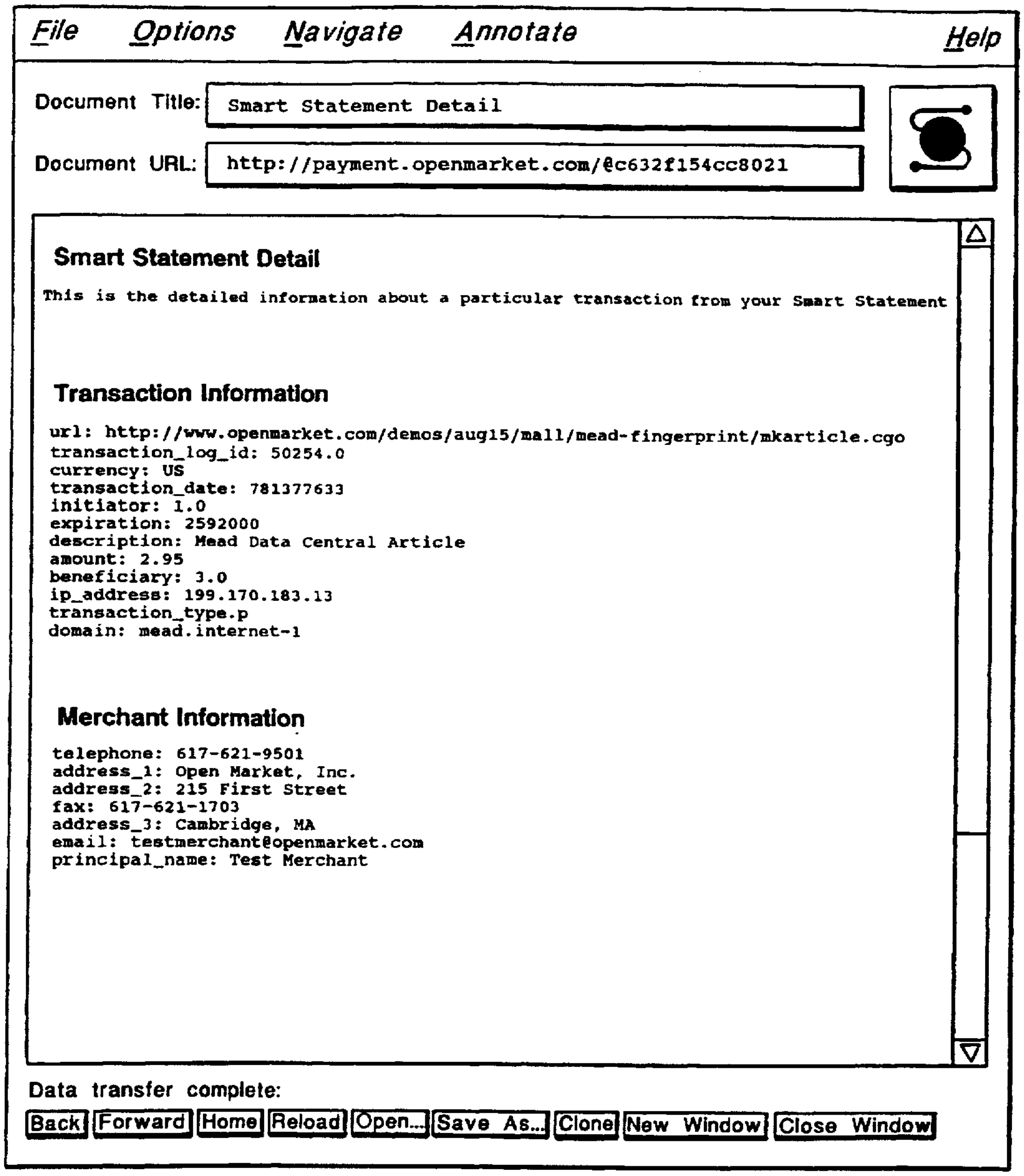
- September 1994
- August 1994

Return to your Newest Statement

Feedback

You can send us comments and suggestions here.

FIG. 11



Document Title: Smart Statement Detail

Document URL: http://payment.openmarket.com/@c632f154cc8021

Smart Statement Detail

This is the detailed information about a particular transaction from your Smart Statement

Transaction Information

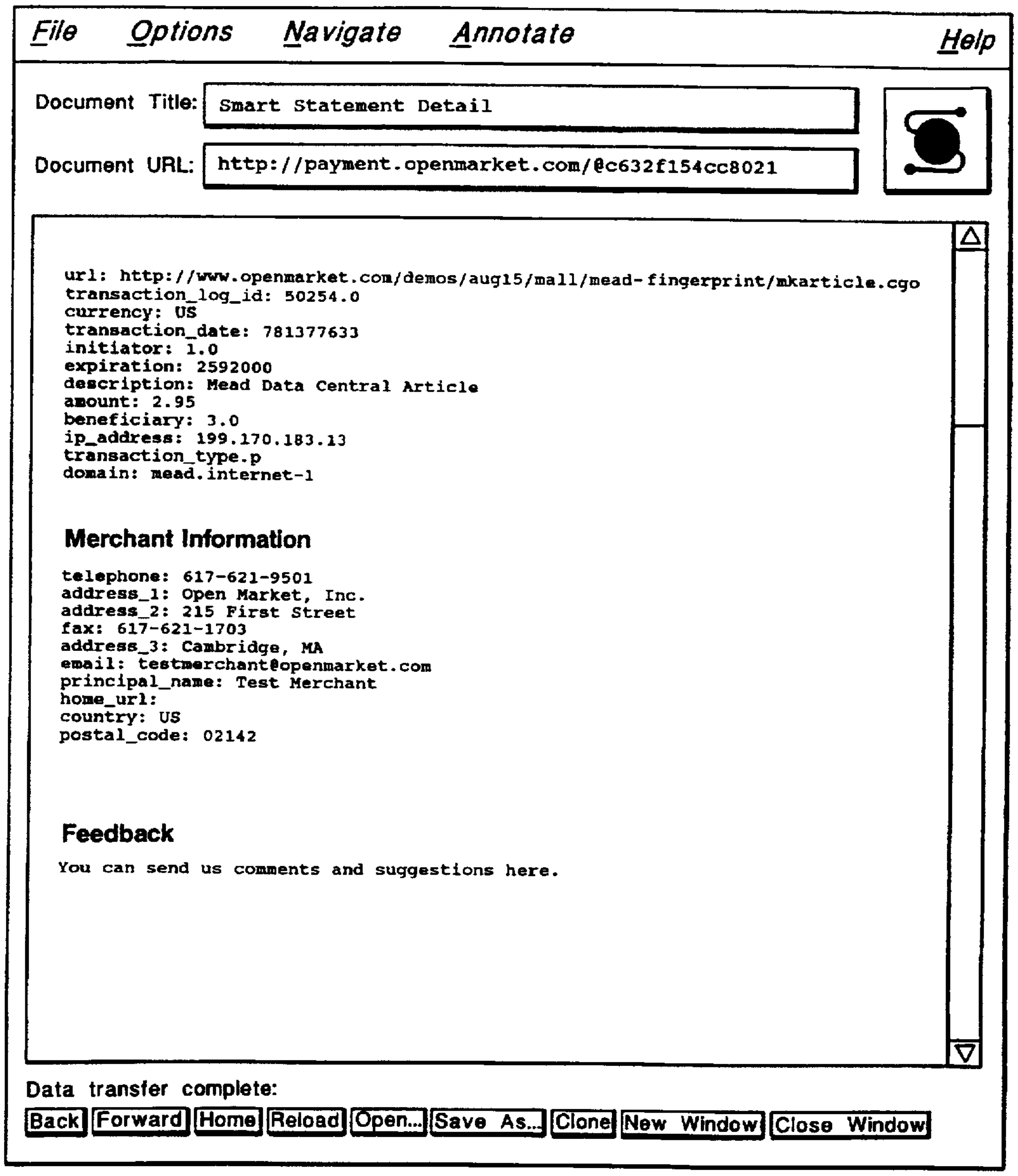
```
url: http://www.openmarket.com/demos/aug15/mall/mead-fingerprint/mkarticle.cgo
transaction_log_id: 50254.0
currency: US
transaction_date: 781377633
initiator: 1.0
expiration: 2592000
description: Mead Data Central Article
amount: 2.95
beneficiary: 3.0
ip_address: 199.170.183.13
transaction_type.p
domain: mead.internet-1
```

Merchant Information

```
telephone: 617-621-9501
address_1: Open Market, Inc.
address_2: 215 First Street
fax: 617-621-1703
address_3: Cambridge, MA
email: testmerchant@openmarket.com
principal_name: Test Merchant
```

Data transfer complete:

FIG. 12



Document Title: Smart Statement Detail

Document URL: http://payment.openmarket.com/@c632f154cc8021

```
url: http://www.openmarket.com/demos/aug15/mall/mead-fingerprint/mkarticle.cgo
transaction_log_id: 50254.0
currency: US
transaction_date: 781377633
initiator: 1.0
expiration: 2592000
description: Mead Data Central Article
amount: 2.95
beneficiary: 3.0
ip_address: 199.170.183.13
transaction_type.p
domain: mead.internet-1
```

Merchant Information

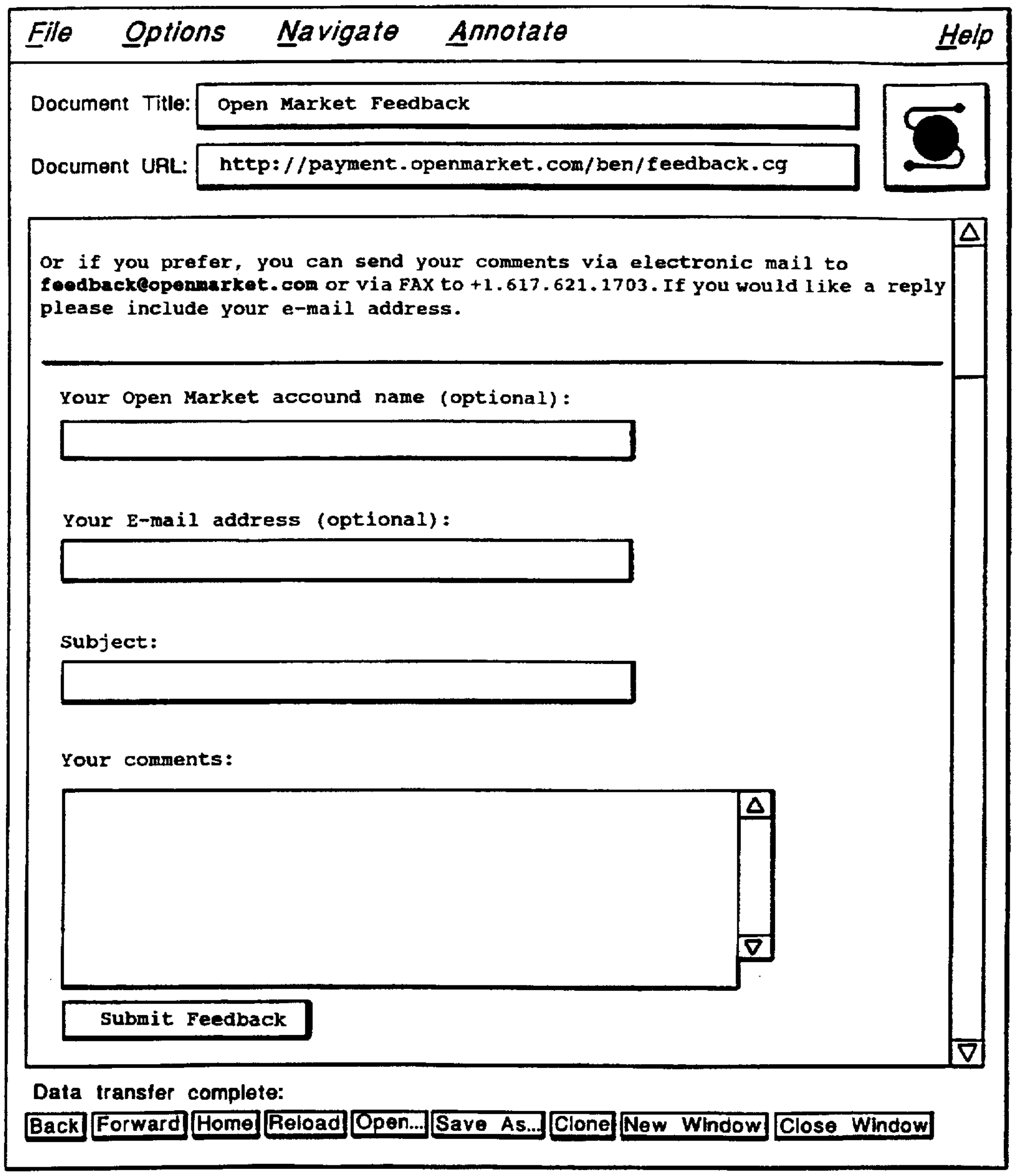
```
telephone: 617-621-9501
address_1: Open Market, Inc.
address_2: 215 First Street
fax: 617-621-1703
address_3: Cambridge, MA
email: testmerchant@openmarket.com
principal_name: Test Merchant
home_url:
country: US
postal_code: 02142
```

Feedback

You can send us comments and suggestions here.

Data transfer complete:

FIG. 13



Document Title: Open Market Feedback

Document URL: http://payment.openmarket.com/ben/feedback.cg

Or if you prefer, you can send your comments via electronic mail to **feedback@openmarket.com** or via FAX to +1.617.621.1703. If you would like a reply please include your e-mail address.

Your Open Market accound name (optional):

Your E-mail address (optional):

Subject:

Your comments:

Data transfer complete:

FIG. 14

NETWORK SALES SYSTEM

CROSS REFERENCE TO RELATED APPLICATION

This is a continuation of U.S. patent application Ser. No. 08/328,133, filed Oct. 24, 1994, now U.S. Pat. No. 5,715, 314.

REFERENCE TO MICROFICHE APPENDICES

Microfiche Appendices A–G are being submitted with the present application, being 4 sheets with 220 total pages.

BACKGROUND OF THE INVENTION

This invention relates to user-interactive network sales systems for implementing an open marketplace for goods or services over computer networks such as the Internet.

U.S. patent application Ser. No. 08/168,519, filed Dec. 16, 1993 by David K. Gifford and entitled "Digital Active Advertising," now abandoned, the entire disclosure of which is hereby incorporated herein in its entirety by reference, describes a network sales system that includes a plurality of buyer computers, a plurality of merchant computers, and a payment computer. A user at a buyer computer asks to have advertisements displayed, and the buyer computer requests advertisements from a merchant computer, which sends the advertisements to the buyer computer. The user then requests purchase of an advertised product, and the buyer computer sends a purchase message to the merchant computer. The merchant computer constructs a payment order that it sends to the payment computer, which authorizes the purchase and sends an authorization message to the merchant computer. When the merchant computer receives the authorization message it sends the product to the buyer computer.

The above-mentioned patent application also describes an alternative implementation of the network sales system in which, when the user requests purchase of an advertised product, the buyer computer sends a payment order directly to the payment computer, which sends an authorization message back to the buyer computer that includes an unforgeable certificate that the payment order is valid. The buyer computer then constructs a purchase message that includes the unforgeable certificate and sends it to the merchant computer. When the merchant computer receives the purchase request it sends the product to the buyer computer, based upon the pre-authorized payment order.

SUMMARY OF THE INVENTION

In one aspect, the invention provides a network-based sales system that includes at least one buyer computer for operation by a user desiring to buy a product, at least one merchant computer, and at least one payment computer. The buyer computer, the merchant computer, and the payment computer are interconnected by a computer network. The buyer computer is programmed to receive a user request for purchasing a product, and to cause a payment message to be sent to the payment computer that comprises a product identifier identifying the product. The payment computer is programmed to receive the payment message, to cause an access message to be created that comprises the product identifier and an access message authenticator based on a cryptographic key, and to cause the access message to be sent to the merchant computer. The merchant computer is programmed to receive the access message, to verify the access message authenticator to ensure that the access

message authenticator was created using the cryptographic key, and to cause the product to be sent to the user desiring to buy the product.

The invention provides a simple design architecture for the network sales system that allows the merchant computer to respond to payment orders from the buyer computer without the merchant computer having to communicate directly with the payment computer to ensure that the user is authorized to purchase the product and without the merchant computer having to store information in a database regarding which buyers are authorized to purchase which products. Rather, when the merchant computer receives an access message from the buyer computer identifying a product to be purchased, the merchant computer need only check the access message to ensure that it was created by the payment computer (thereby establishing for the merchant computer that the buyer is authorized to purchase the product), and then the merchant computer can cause the product to be sent to the buyer computer who has been authorized to purchase the product.

In another aspect, the invention features a network-based sales system that includes at least one buyer computer for operation by a user desiring to buy products, at least one shopping cart computer, and a shopping cart database connected to the shopping cart computer. The buyer computer and the shopping cart computer are interconnected by a computer network. The buyer computer is programmed to receive a plurality of requests from a user to add a plurality of respective products to a shopping cart in the shopping cart database, and, in response to the requests to add the products, to send a plurality of respective shopping cart messages to the shopping cart computer each of which includes a product identifier identifying one of the plurality of products. The shopping cart computer is programmed to receive the plurality of shopping cart messages, to modify the shopping cart in the shopping cart database to reflect the plurality of requests to add the plurality of products to the shopping cart, and to cause a payment message associated with the shopping cart to be created. The buyer computer is programmed to receive a request from the user to purchase the plurality of products added to the shopping cart and to cause the payment message to be activated to initiate a payment transaction for the plurality of products added to the shopping cart.

In another aspect, the invention features a network-based link message system that includes at least one client computer for operation by a client user and at least one server computer for operation by a server user. The client computer and the server computer are interconnected by a computer network. The client computer is programmed to send an initial link message to the server computer. The server computer is programmed to receive the initial link message from the client computer and to create, based on information contained in the initial link message, a session link message that encodes a state of interaction between the client computer and the server computer. The session link message includes a session link authenticator, computed by a cryptographic function of the session link contents, for authenticating the session link message. The server computer is programmed to cause the session link message to be sent to the client computer. The client computer is programmed to cause the session link message to be sent to a computer in the network that is programmed to authenticate the session link message by examining the session link authenticator and that is programmed to respond to the session link message based on the state of the interaction between the client computer and the server computer.

In another aspect, the invention features a network-based sales system that includes a merchant database having a plurality of digital advertisements and a plurality of respective product fulfillment items, at least one creation computer for creating the merchant database, and at least one merchant computer for causing the digital advertisements to be transmitted to a user and for causing advertised products to be transmitted to the user. The creation computer and the merchant computer are interconnected by a computer network. The creation computer is programmed to create the merchant database, and to transmit the digital advertisements and the product fulfillment items to the merchant computer. The merchant computer is programmed to receive the digital advertisements and product fulfillment items, to receive a request for a digital advertisement from a user, to cause the digital advertisement to be sent to the user, to receive from the user an access message identifying an advertised product, and to cause the product to be sent to the user in accordance with a product fulfillment item corresponding to the product.

In another aspect, the invention features a hypertext statement system that includes a client computer for operation by a client user and one or more server computers for operation by a server user. The client computer and the server computers are interconnected by a computer network. At least one of the server computers is programmed to record purchase transaction records in a database. Each of the purchase transaction records includes a product description. The server computer is programmed to transmit a statement document that includes the purchase transaction records to the client computer. The client computer is programmed to display the product descriptions, to receive a request from the client user to display a product corresponding to a product description displayed by the client computer, and to cause a product hypertext link derived from a purchase transaction record to be activated. At least one of the server computers is programmed to respond to activation of the product hypertext link by causing the product to be sent to the client computer.

In another aspect, the invention features a network payment system that includes at least one buyer computer for operation by a user desiring to buy a product and at least one payment computer for processing payment messages from the buyer computer. The buyer computer and the payment computer are interconnected by a computer network. The buyer computer is programmed to cause a payment message to be sent to the payment computer. The payment message includes a product identifier identifying the product that the user desires to buy. The payment computer is programmed to receive the payment message, to cause an access message to be created to enable the user to access the product, and to record a purchase transaction record in the settlement database. The buyer computer is programmed to cause a request for purchase transaction records to be sent to the payment computer. The payment computer is programmed to receive the request for purchase transaction records and to cause a document derived from the purchase transaction records to be sent to the buyer computer.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram of a network sales system in accordance with the present invention.

FIG. **2** (**2**-A through **2**-I) is a flowchart diagram illustrating the operation of a purchase transaction in the network sales system of FIG. **1**.

FIG. **3** (**3**-A through **3**-B) is a flowchart diagram illustrating the use of a shopping cart for the purchase of products in connection with the network sales system of FIG. **1**.

FIG. **4** (**4**-A through **4**-C) is a flowchart diagram illustrating the operation of a smart statement in the network sales system of FIG. **1**.

FIG. **5** is a screen snapshot of an advertising document that the merchant computer sends to the buyer computer in FIG. **2**.

FIG. **6** is a screen snapshot of a confirmation document that the payment computer sends to the buyer computer in FIG. **2**.

FIG. **7** is a screen snapshot of a new account document that the payment computer sends to the buyer computer in FIG. **2**.

FIG. **8** is a screen snapshot of an account name prompt that the buyer computer creates in FIG. **2**.

FIG. **9** is a screen snapshot of a document that the payment computer sends to the buyer computer in FIG. **2** and that provides an option either to repurchase or to use a previously purchased access.

FIG. **10** is a screen snapshot of a fulfillment document that the merchant computer sends to the buyer computer in FIG. **2**.

FIG. **11** is a screen snapshot of a smart statement document that the payment computer sends to the buyer computer in FIG. **4**.

FIGS. **12** and **13** are screen snapshots of a transaction detail document that the payment computer sends to the buyer computer in FIG. **4**.

FIG. **14** is a screen snapshot of a customer service form that the payment computer sends to the buyer computer in FIG. **4**.

DETAILED DESCRIPTION

With reference to FIG. **1**, a network sales system in accordance with the present invention includes a buyer computer **12** operated by a user desiring to buy a product, a merchant computer **14**, which may be operated by a merchant willing to sell products to the buyer or by a manager of the network sales system, a payment computer **16** typically operated by a manager of the network sales system, and a creation computer **20** typically operated by the merchant. The buyer, merchant, payment, and creation computers are all inter-connected by a computer network **10** such as the Internet.

Creation computer **20** is programmed to build a "store" of products for the merchant. A printout of a computer program for use in creating such a "store" in accordance with the present invention is provided as Appendix F.

The products advertised by merchant computer **14** may be, for example, newspaper or newsletter articles available for purchase by buyers. Creation computer **20** creates a digital advertisement database **18** that stores advertising documents (which may for example be in the form of summaries of newspaper or newsletter articles, accompanied by prices) and product fulfillment items (which may be the products themselves if the products can be transmitted over the network, or which may be hard goods identifiers if the products are hard goods, i.e., durable products as opposed to information products). Creation computer **20** transmits contents of the advertising document database **18** to merchant computer **14** to enable the merchant computer to cause advertisements and products to be sent to buyers. Merchant computer **14** maintains advertising documents locally in advertising document database **15**. In an alternative embodiment, the creation computer does not have a local digital advertisement database, but instead updates a remote advertising document database on a merchant computer. These updates can be accomplished using HTML forms or other remote database technologies as is understood by practitioners of the art.

Payment computer **16** has access to a settlement database **22** in which payment computer **16** can record details of purchase transactions. The products may be organized into various "domains" of products, and payment computer **16** can access settlement database **22** to record and retrieve records of purchases of products falling within the various domains. Payment computer **16** also has access to a shopping cart database **21** in which a "shopping cart" of products that a user wishes to purchase can be maintained as the user shops prior to actual purchase of the contents of the shopping cart.

With reference to FIG. **2**, a purchase transaction begins when a user at buyer computer **12** requests advertisements (step **24**) and buyer computer **12** accordingly sends an advertising document URL (universal resource locator) to merchant computer **14** (step **26**). The merchant computer fetches an advertising document from the advertising document database (step **28**) and sends it to the buyer computer (step **30**). An example of an advertising document is shown in FIG. **5**. Details of URLs and how they are used are found in Appendix G.

The user browses through the advertising document and eventually requests a product (step **32**). This results in the buyer computer sending payment URL A to the payment computer (step **34**). Payment URL A includes a product identifier that represents the product the user wishes to buy, a domain identifier that represents a domain of products to which the desired product belongs, a payment amount that represents the price of the product, a merchant computer identifier that represents merchant computer **14**, a merchant account identifier that represents the particular merchant account to be credited with the payment amount, a duration time that represents the length of time for which access to the product is to be granted to the user after completion of the purchase transaction, an expiration time that represents a deadline beyond which this particular payment URL cannot be used, a buyer network address, and a payment URL authenticator that is a digital signature based on a cryptographic key. The payment URL authenticator is a hash of other information in the payment URL, the hash being defined by a key shared by the merchant and the operator of the payment computer.

In an alternative embodiment, step **34** consists of the buyer computer sending a purchase product message to the merchant computer, and the merchant computer provides payment URL A to the buyer computer in response to the purchase product message. In this alternative embodiment, payment URL A contains the same contents as above. The buyer computer then sends the payment URL A it has received from the merchant computer to the payment computer.

When the payment computer receives the payment URL it verifies whether the payment URL authenticator was created from the contents of the payment URL using the cryptographic key (step **36**). If not, the payment computer sends a document to the buyer computer indicating that access to the network sales system is denied (step **38**). Otherwise, the payment computer determines whether the expiration time has past (step **40**). If it has, the payment computer sends a document to the buyer computer indicating that the time has expired (step **41**). Otherwise, the payment computer checks the buyer computer network

address to see if it matches the one specified in the payment URL (step **42**). If it does not match, the payment computer sends a document to the buyer computer indicating that access to the network payment system is denied (step **43**). Otherwise, the payment computer sends a payment confirmation document to the buyer computer, the payment confirmation document including an "open" link and a "continue" link (step **44**).

An example of a confirmation document is shown in FIG. **6**. The confirmation document asks the user to click on a "continue" button if the user already has an account with the payment computer, or to click on an "open" button if the user does not already have an account and wishes to open one.

If the user clicks on the "open" button (step **46**), the buyer computer sends payment URL C to the payment computer (step **48**), payment URL C being similar to payment URL A but also indicating that the user does not yet have an account. The payment computer creates a new account document (step **50**) and sends it to the buyer computer (step **52**). An example of a new account document is shown in FIG. **7**. When the user receives the new account document he enters the new account name, an account password, a credit card number, the credit card expiration date, and security information such as the maiden name of the user's mother (step **54**), and presses a "submit" button (not shown in FIG. **7**). The buyer computer sends the new account information to the payment computer (step **56**), which enters the new account in the settlement database (step **58**).

If the user clicks on the "continue" button (step **60**), the buyer computer sends payment URL B to the payment computer (step **62**), payment URL B being similar to payment URL A but also indicating that the user already has an account. The payment computer then instructs the buyer computer to provide the account name and password (steps **64** and **66**), and the buyer computer prompts the user for this information by creating an account name prompt (example shown in FIG. **8**) and a similar password prompt. The user enters the information (step **68**) and the buyer computer sends the account name and password to the payment computer (step **70**).

The payment computer verifies whether the user name and password are correct (step **72**). If they are not correct, the payment computer sends a document to the buyer computer indicating that access to the network sales system is denied (step **74**). Otherwise, the payment computer determines whether additional security is warranted, based on, e.g., whether the payment amount exceeds a threshold (step **73**). If additional security is warranted, the payment computer creates a challenge form document and sends it to the buyer computer (step **75**). The user enters the security information (step **77**), the buyer computer sends the security information to the payment computer (step **79**), and the payment computer determines whether the security information is correct (step **81**). If it is not correct, the payment computer sends a document to the buyer computer indicating that access to the network sales system is denied (step **83**).

If the security information is correct, or if additional security was not warranted, the payment computer checks the settlement database to determine whether the user has unexpired access to the domain identifier contained in the payment URL (step **82**). If so, the payment computer sends to the buyer computer a document providing an option either to repurchase or to use the previously purchased access (step **84**). An example of such a document is shown in FIG. **9**. The user can respond to the recent purchase query document by choosing to access the previously purchased document (step **85**) or to go ahead and buy the currently selected product (step **86**).

If the user chooses to access the previously purchased document, the buyer computer skips to step **92** (see below). If the user chooses to buy the currently selected product, the payment computer calculates an actual payment amount that may differ from the payment amount contained in the payment URL (step **87**). For example, the purchase of a product in a certain domain may entitle the user to access other products in the domain for free or for a reduced price for a given period of time.

The payment computer then verifies whether the user account has sufficient funds or credit (step **76**). If not, the payment computer sends a document to the buyer computer indicating that the user account has insufficient funds (step **78**). Otherwise, the payment computer creates an access URL (step **80**) that includes a merchant computer identifier, a domain identifier, a product identifier, an indication of the end of the duration time for which access to the product is to be granted, the buyer network address, and an access URL authenticator that is a digital signature based on a cryptographic key. The access URL authenticator is a hash of other information in the access URL, the hash being defined by a key shared by the merchant and the operator of the payment computer. The payment computer then records the product identifier, the domain, the user account, the merchant account, the end of duration time, and the actual payment amount in the settlement database (step **88**).

The payment computer then sends a redirect to access URL to the buyer computer (step **90**), which sends the access URL to the merchant computer (step **92**). The merchant computer verifies whether the access URL authenticator was created from the contents of the access URL using the cryptographic key (step **94**). If not, the merchant computer sends a document to the buyer computer indicating that access to the product is denied (step **96**).

Otherwise, the merchant computer verifies whether the duration time for access to the product has expired (step **98**). This is done because the buyer computer can request access to a purchased product repeatedly. If the duration time has expired, the merchant computer sends a document to the buyer computer indicating that the time has expired (step **100**). Otherwise the merchant computer verifies that the buyer computer network address is the same as the buyer network address in the access URL (step **101**), and if so, sends a fulfillment document to the buyer computer (step **102**), which is displayed by the buyer computer (step **104**). An example of a fulfillment document is shown in FIG. **10**. Otherwise, the merchant computer sends a document to the buyer computer indicating that access is not allowed (step **103**).

With reference now to FIG. **3**, when the merchant computer sends the advertising document to the buyer computer, the user may request that a product be added to a shopping cart in the shopping cart database rather than request that the product be purchased immediately. The buyer computer sends a shopping cart URL to the payment computer (step **108**), the shopping cart URL including a product identifier, a domain identifier, a payment amount, a merchant computer identifier, a merchant account identifier, a duration time, an expiration time, and a shopping cart URL authenticator that is a digital signature based on a cryptographic key. The shopping cart URL authenticator is a hash of other information in the shopping cart URL, the hash being defined by a key shared by the merchant and the operator of the payment computer.

The payment computer verifies whether the shopping cart URL authenticator was created from the contents of the shopping cart URL using a cryptographic key (step **110**). If not, the payment computer sends a document to the buyer computer indicating that access to the network sales system is denied (step **112**). Otherwise, before any modification to a user's shopping cart is allowed, user authentication is performed (step **113**) in a manner analogous to steps **40**–**81**. Once the user is authenticated, the payment computer creates or updates a payment URL for the shopping cart (step **114**).

The user then either requests more advertisements (step **24** in FIG. **2**) and possibly adds another product to the shopping cart, requests display of the shopping cart (step **116**), or requests purchase of the entire contents of the shopping cart (step **124**). If the user requests display of the shopping cart (step **116**), the buyer computer sends a fetch shopping cart request to the payment computer (step **118**), and the payment computer and buyer computer (step **119**) perform steps analogous to steps **64**–**81**. The payment computer returns the contents of the shopping cart to the buyer computer (step **120**), which displays the contents of the shopping cart (step **122**). If the user requests that the entire contents of the shopping cart be purchased (step **124**) the buyer computer causes the payment URL for the shopping cart to be activated (step **126**) and the payment URL is processed in a manner analogous to the processing of payment URLs for individual products (beginning with step **36** in FIG. **2**).

With reference now to FIG. **4**, a user can request display of a "smart statement" that lists purchase transactions for a given month (step **128**). When the buyer computer receives such a request, it sends a smart statement URL to the payment computer (step **130**).

When the payment computer receives the smart statement URL, it verifies whether the smart statement URL authenticator was created from the contents of the smart statement URL using a cryptographic key (step **132**). If not, the payment computer sends a document to the buyer computer indicating that access is denied (step **134**). Otherwise, the payment computer checks to determine whether the buyer network address in the smart statement URL matches the buyer computer's actual network address (step **136**). If not, the payment computer sends a document to the buyer computer indicating that access is denied (step **138**). Otherwise (step **140**), the payment computer and buyer computer perform a set of steps analogous to steps **64**–**81** in FIG. **2** (payment computer requests account name and password, user provides the requested information, and payment computer verifies the information).

In an alternative embodiment steps **132**–**138** are omitted.

After verification of account information is complete, the payment computer retrieves the requested settlement data from the settlement database, creates a smart statement document for the buyer, and sends the smart statement document to the buyer computer (step **142**). An example of a smart statement document is shown in FIG. **11**. Each purchase transaction record in the smart statement document includes the data of the transaction, the name of the merchant, an identification of the product, and the payment amount for the product. The smart statement document also includes a transaction detail URL for each purchase transaction (these URLs, or hypertext links, are discussed below and are not shown in FIG. **11**). The smart statement document also identifies previous statements that the user may wish to have displayed.

The buyer computer displays the retrieved document (step **144**), and the user may request transaction details for a particular transaction listed on the smart statement (step **146**). If so, the buyer computer sends a transaction detail URL (or "payment detail URL") to the payment computer (step **148**). The transaction detail URL includes a transaction identifier, a buyer network address, and a transaction detail URL authenticator. When the payment computer receives the transaction detail URL, it performs (step **150**) a set of steps analogous to steps **132–140** (verification of URL authenticator, buyer network address, and account information). The payment computer then retrieves from the settlement database data corresponding to the payment transaction specified in the transaction detail URL, creates a transaction detail document, and sends it to the buyer computer (step **152**).

An example of a transaction detail document is shown in FIGS. **12** and **13**. The document displays a number of items of information about the transaction, including the transaction date, end of the duration time ("expiration"), a description of the product, the payment amount, the domain corresponding to the product, an identification of the merchant, and the merchant's address.

The smart statement document and the transaction detail document both include customer service URLs (hypertext links) that allow the user to request customer service (i.e., to send comments and suggestions to the payment computer). When the user requests customer service (step **154**), the buyer computer sends the customer service URL to the payment computer (step **156**), which creates a customer service form and sends it to the buyer computer (step **158**). An example of a customer service form is shown in FIG. **14**. The user types comments into the customer service form (step **160**), and the buyer computer sends the user's comments to the payment computer (step **162**). The payment computer then posts the user comments and sends a thank you document to the buyer computer (step **164**).

A user may request display of a product included in the smart statement. When the user requests that the product be displayed (step **166**), the buyer computer sends the access URL contained in the smart statement document to the merchant computer (step **168**), and the buyer computer and merchant computer perform a set of steps analogous to steps **94–104** in FIG. **2** (authentication of access URL, verification whether duration time has expired, verification of buyer network address, and transmission of fulfillment document to buyer computer).

Whenever the present application states that one computer sends a URL to another computer, it should be understood that in preferred embodiments the URL is sent in a standard HTTP request message, unless a URL message is specified as a redirection in the present application. The request message includes components of the URL as described by the standard HTTP protocol definition. These URL components in the request message allow the server to provide a response appropriate to the URL. The term "URL" as used the present application is an example of a "link," which is a pointer to another document or form (including multimedia documents, hypertext documents including other links, or audio/video documents).

When the present application states that one computer sends a document to another computer, it should be understood that in preferred embodiments the document is a success HTTP response message with the document in the body of the message. When the present application states that a server sends an account name and password request message to the client, it should be understood that in preferred embodiments the account name and password request message is an unauthorized HTTP response. A client computer sends account name and password information to a server as part of a request message with an authorization field.

The software architecture underlying the particular preferred embodiment is based upon the hypertext conventions of the World Wide Web. Appendix A describes the Hypertext Markup Language (HTML) document format used to represent digital advertisements, Appendix B describes the HTML forms fill out support in Mosaic **2.0**, Appendix C is a description of the Hypertext Transfer Protocol (HTTP) between buyer and merchant computers, Appendix D describes how documents are named with Uniform Resource Locators (URLs) in the network of computers, and Appendix E describes the authentication of URLs using digital signatures.

A printout of a computer program for use in creating and operating such a "store" in accordance with the present invention is provided as Appendix F. A printout of a computer program for use in operating other aspects of the network sales system in accordance with the present invention is provided in Appendix G.

There has been described a new and useful network-based sales system. It is apparent that those skilled in the art may make numerous modifications and departures from the specific embodiments described herein without departing from the spirit and scope of the claimed invention.

What is claimed is:

1. A network-based sales system, comprising:

a merchant database comprising a plurality of digital advertisements and a plurality of respective product fulfillment items;

at least one creation computer for creating said merchant database; and

at least one merchant computer for causing said digital advertisements to be transmitted to a user and for causing advertised products to be transmitted to said user;

said creation computers, said merchant computer, and a payment computer being interconnected by a public packet switched computer network;

said creation computer being programmed to create said merchant database, and to transmit said digital advertisements and said product fulfillment items over said network to said merchant computer;

said merchant computer being programmed to receive said digital advertisements and product fulfillment items over said network, to receive over said network a request for a digital advertisement from a user, to cause said digital advertisement to be sent to said user over said network, to receive over said network from said user a product request message identifying an advertised product, to receive an access message over said network created by said payment computer, and to cause said product to be sent to said user in accordance with a product fulfillment item corresponding to said product and based upon receipt by the merchant computer of the access message.

2. A network-based sales system in accordance with claim **1**, wherein each of said digital advertisements comprises an abstract of a product and a price.

3. A network-based sales system in accordance with claim **2**, wherein:

at least one of said product fulfillment items comprises a product itself; and

said creation computer is programmed to transmit said product to said merchant computer with said digital advertisements.

4. A network-based sales system in accordance with claim 2, wherein:

at least one of said product fulfillment items comprises a hard good identifier; and

said creation computer is programmed to transmit said hard good identifier to said merchant computer with said digital advertisements.

5. A method of operating a merchant computer in a network-based sales system comprising a merchant database that comprises a plurality of digital advertisements and a plurality of respective product fulfillment items, at least one creation computer for creating said merchant database, and at least one merchant computer for causing said digital advertisements to be transmitted to a user and for causing advertised products to be transmitted to said user, and at least one payment computer, said creation computer, said merchant computer, and said payment computer being interconnected by a public packet switched computer network, said method comprising the steps of:

receiving, at said merchant computer, said digital advertisements and said product fulfillment items, said digital advertisements and said product fulfillment items having been transmitted over said network to said merchant computer by said creation computer, said merchant database comprising said digital advertisements and said product fulfillment items having been created by said creation computer;

receiving over said network a request for a digital advertisement from a user;

causing said digital advertisement to be sent to said user over said network;

receiving over said network from said user a product request message identifying an advertised product;

receiving over said network an access message created by said payment computer; and

causing said product to be sent to said user in accordance with a product fulfillment item corresponding to said product and based upon receipt by the merchant computer of the access message.

6. A hypertext statement system, comprising:

a client computer for operation by a client user; and

a plurality of server computers for operation by a server user;

said client computer and said server computers being interconnected by a public packet switched computer network;

at least one of said server computers being programmed to record information pertaining to purchase transaction records in a database, each of said purchase transaction records comprising a product description, and to cause a statement document comprising said purchase transaction records to be transmitted to said client computer over said network;

said client computer being programmed to display said product descriptions, to receive a request from said client user to display a product corresponding to a product description displayed by said client computer, and to cause a product hypertext link derived from a purchase transaction record to be activated;

at least one of said server computers, other than a server computer that transmitted said statement document to

said client computer, being programmed to respond to activation of said product hypertext link by causing said product to be sent to said client computer over said network.

7. A hypertext statement system in accordance with claim 6, wherein:

said client computer is programmed to receive a request from said client user to display transaction details corresponding to a product description displayed by said client computer and to cause a transaction detail hypertext link corresponding to said product description to be activated; and

at least one of said server computers is programmed to respond to activation of said transaction detail hypertext link by transmitting said transaction details to said client computer as a transaction detail document.

8. A hypertext statement system in accordance with claim 7, wherein:

said transaction detail document further comprises a customer service form hypertext link;

said client computer is programmed to receive a request from said client user to display a customer service form and to cause said customer service form hypertext link to be activated; and

at least one of said server computers is programmed to respond to activation of said customer service form hypertext link by transmitting said customer service form to said client computer.

9. A hypertext statement system in accordance with claim 6, wherein:

said statement document further comprises a customer service form hypertext link;

said client computer is programmed to receive a request from said client user to display a customer service form and to cause said customer service form hypertext link to be activated; and

at least one of said server computers is programmed to respond to activation of said customer service form hypertext link by transmitting said customer service form to said client computer.

10. A method of operating a server computer in a hypertext statement system comprising a client computer for operation by a client user, and a plurality of server computers for operation by a server user, said client computer and said server computers being interconnected by a public packet switched computer network, said method comprising the steps of:

recording, at one of said server computers, information pertaining to purchase transaction records in a database, each of said purchase transaction records comprising a product description; and

causing a statement document comprising said purchase transaction records to be transmitted to said client computer over said network;

said client computer being programmed to display said product descriptions, to receive a request from said client user to display a product corresponding to a product description displayed by said client computer, and to cause a product hypertext link derived from a purchase transaction record to be activated;

at least one of said server computers, other than a server computer that transmitted said statement document to said client computer, being programmed to respond to activation of said product hypertext link by causing said product to be sent to said client computer over said network.

11. A network payment system, comprising:

at least one buyer computer for operation by a user desiring to buy a product; and

at least one payment computer for processing payment messages from said buyer computer;

said buyer computer, said payment computer, and a merchant computer being interconnected by a public packet switched computer network;

said buyer computer being programmed to cause a payment message to be sent to said payment computer over said network;

said payment computer being programmed to receive said payment message, to cause an access message to be created for transmission over said network to said merchant computer to enable said user to access said product upon verification by said merchant computer that said access message was created by said payment computer, and to record information pertaining to a purchase transaction record in said settlement database;

said buyer computer being programmed to cause a request for a purchase transaction record to be sent to said payment computer over said network; and

said payment computer being programmed to receive said request for said purchase transaction record and to cause a document derived from said purchase transaction record to be sent to said buyer computer over said network.

12. The network payment system of claim 11 wherein the payment message comprises a product identifier identifying the product that the user desires to buy.

13. A method of operating a payment computer in a network payment system comprising at least one buyer computer for operation by a user desiring to buy a product, and at least one payment computer for processing payment messages from said buyer computer, and at least one merchant computer, said buyer computer, said payment computer, and said merchant computer being interconnected by a public packet switched computer network, said method comprising the steps of:

receiving, at said payment computer, a payment message that said buyer computer has caused to be sent to said payment computer over said network;

causing an access message to be created for transmission to a merchant computer over said network to enable said user to access said product upon verification by said merchant computer that said access message was created by said payment computer;

recording information pertaining to a purchase transaction record in said settlement database;

receiving over said network a request for a purchase transaction record that said buyer computer has caused to be sent to said payment computer; and

causing a document derived from said purchase transaction record to be sent to said buyer computer over said network.

14. The method of claim 13 wherein the payment message comprises a product identifier identifying the product that the user desires to buy.

15. A hypertext statement system, comprising:

a client computer for operation by a client user; and

one or more server computers for operation by a server user;

the client computer and the server computers being interconnected by a public packet switched computer network;

at least one of the server computers being programmed to record information pertaining to purchase transaction records in a database, and to transmit a statement document comprising the purchase transaction records to the client computer over the network;

the client computer being programmed to display the statement document to receive a request from the client user to display transaction details corresponding to a portion of the statement document displayed by the client computer, and to cause a transaction detail hypertext link corresponding to the portion of the statement document to be activated;

at least one of the server computers being programmed to respond to activation of the transaction detail hypertext link by transmitting the transaction details to the client computer over the network as a transaction detail document.

16. A method of operating a server computer in a hypertext statement system comprising a client computer for operation by a client user, and one or more server computers for operation by a server user, the client computer and the server computers being interconnected by a public packet switched computer network, the method comprising the steps of:

recording, at one of the server computers, information pertaining to purchase transaction records in a database; and

transmitting a statement document comprising the purchase transaction records to the client computer over the network;

the client computer being programmed to display the statement document, to receive a request from the client user to display transaction details corresponding to a portion of the statement document displayed by the client computer, and to cause a transaction detail hypertext link corresponding to the portion of the statement document to be activated;

at least one of the server computers being programmed to respond to activation of the transaction detail hypertext link by transmitting the transaction details to the client computer over the network as a transaction detail document.

17. A network-based sales system, comprising:

at least one buyer computer for operation by a user desiring to buy products;

at least one shopping cart computer; and

a shopping cart database connected to the shopping cart computer;

the buyer computer and the shopping cart computer being interconnected by a public packet switched computer network;

the buyer computer being programmed to receive a plurality of requests from a user to add a plurality of respective products to a shopping cart in the shopping cart database, and, in response to the requests to add the products, to send a plurality of respective shopping cart messages over the network to the shopping cart computer each of which comprises a product identifier identifying one of the plurality of products and at least one of which comprises a universal resource locator;

the shopping cart computer being programmed to receive the plurality of shopping cart messages, to modify the shopping cart in the shopping cart database to reflect the plurality of requests to add the plurality of products to the shopping cart, and to cause a payment message associated with the shopping cart to be created, the payment message comprising a universal resource locator; and

the buyer computer being programmed to receive a request from the user to purchase the plurality of products added to the shopping cart and to cause the payment message to be activated to initiate a payment transaction for the plurality of products added to the shopping cart;

the shopping cart being a stored representation of a collection of products, the shopping cart database being a database of stored representations of collections of products, and the shopping cart computer being a computer that modifies the stored representations of collections of products in the database.

18. A method of operating a shopping cart computer in a public packet switched computer network comprising at least one buyer computer for operation by a user desiring to buy products, at least one shopping cart computer, and a shopping cart database connected to the shopping cart computer, the method comprising the steps of:

receiving, at the shopping cart computer, a plurality of shopping cart messages sent over the network to the shopping cart computer by the buyer computer in response to receipt of a plurality of requests from a user to add a plurality of respective products to a shopping cart in the shopping cart database, each of the shopping cart messages comprising a product identifier identifying one of the plurality of products and at least one of which comprises a universal resource locator;

modifying the shopping cart in the shopping cart database to reflect the plurality of requests to add the plurality of products to the shopping cart; and

causing a payment message associated with the shopping cart to be created, the payment message comprising a universal resource locator;

the buyer computer being programmed to receive a request from the user to purchase the plurality of products added to the shopping cart and to cause the payment message to be activated to initiate a payment transaction for the plurality of products added to the shopping cart;

the shopping cart being a stored representation of a collection of products, the shopping cart database being a database of stored representations of collections of products, and the shopping cart computer being a computer that modifies the stored representations of collections of products in the database.

19. A network-based sales system, comprising:

at least one buyer computer for operation by a user desiring to buy a product;

at least one merchant computer; and

at least one payment computer;

the buyer computer, the merchant computer, and the payment computer being interconnected by a computer network;

the buyer computer being programmed to receive a user request for purchasing a product, and to cause a payment message to be sent to the payment computer that comprises a product identifier identifying the product;

the payment computer being programmed to receive the payment message, to cause an access message to be created that comprises a product identifier identifying the product and an access message authenticator based on a cryptographic key, and to cause the access message to be sent to the merchant computer; and

the merchant computer being programmed to receive the access message, to cause the access message authenticator to be verified to ensure that the access message authenticator was created using the cryptographic key, and to cause the product to be received by the user desiring to buy the product.

20. A network-based sales system in accordance with claim 19 wherein the buyer computer is programmed to cause the payment message to be sent to the payment computer by sending a purchase product message to the merchant computer, the merchant computer being programmed to receive the purchase product message, and in response thereto, to send the payment message to the payment computer.

21. A network-based sales system in accordance with claim 19 wherein the merchant computer is programmed itself to verify the access message authenticator.

22. A network-based sales system in accordance with claim 19 wherein the merchant computer is programmed to cause every access message authenticator received by the merchant computer to be verified.

23. A network-based sales system in accordance with claim 19, wherein the payment message comprises a payment amount.

24. A network-based sales system in accordance with claim 19, wherein the payment computer is programmed to record the product identifier and the payment amount.

25. A network-based sales system in accordance with claim 24, wherein the product identifier and the payment amount are recorded in a settlement database.

26. A network-based sales system in accordance with claim 19, wherein the payment message comprises a merchant computer identifier.

27. A network-based sales system in accordance with claim 19, wherein the payment message comprises a payment message authenticator based on a cryptographic key.

28. A network-based sales system in accordance with claim 27, wherein the payment computer is programmed to verify the payment message authenticator to ensure that the payment message authenticator was created using the cryptographic key.

29. A network-based sales system in accordance with claim 19 wherein the computer network is a public packet-switched communications network.

30. A method of operating a payment computer in a computer network comprising at least one buyer computer for operation by a user desiring to buy a product, at least one merchant computer, and at least one payment computer, the method comprising the steps of:

receiving, at the payment computer, a payment message that the buyer computer has caused to be sent to the payment computer in response to a user request for purchasing a product, the payment message comprising a product identifier identifying the product;

causing an access message to be created that comprises a product identifier identifying the product and an access message authenticator based on a cryptographic key; and

causing the access message to be sent to the merchant computer, the merchant computer being programmed to receive the access message, to cause the access message authenticator to be verified to ensure that the access message authenticator was created using the cryptographic key, and to cause the product to be received by the user desiring to buy the product.

31. A network-based sales system, comprising:

at least one buyer computer for operation by a user desiring to buy a product;

at least one merchant computer; and

at least one payment computer;

the buyer computer, the merchant computer, and the payment computer being interconnected by a public packet switched computer network;

the buyer computer being programmed to receive a request from a user for purchasing a product, and to cause a payment message to be sent over the network to the payment computer;

the payment computer being programmed to receive the payment message, and, if purchase of the product by the user has not been previously recorded in a settlement database, to cause the user to be charged for the product and to create a new record in the settlement database reflecting purchase of the product by the user, to cause an access message to be created, and to cause the access message to be sent over the network to the merchant computer; and

the merchant computer being programmed to receive the access message and to cause the user to receive the product.

32. The network-based sales system of claim **31** wherein:

the payment computer is programmed to cause the access message to be created using a cryptographic key; and

at least one of the computers is programmed to use the access message in a cryptographic process to ensure that the user has paid for the product.

33. A method of operating a payment computer in a public packet switched computer network comprising at least one buyer computer for operation by a user desiring to buy a product, at least one merchant computer, and at least one payment computer, the method comprising the steps of:

receiving, at the payment computer, a payment message that the buyer computer has caused to be sent over the network to the payment computer in response to a request from a user for purchasing a product, and, if purchase of the product by the user has not been previously recorded in a settlement database, causing the user to be charged for the product and creating a new record in the settlement database reflecting purchase of the product by the user;

causing an access message to be created; and

causing the access message to be sent over the network to the merchant computer, the merchant computer being programmed to receive the access message and to cause the user to receive the product.

34. The method of claim **33** wherein at least one of the computers is programmed to use the access message in a cryptographic process to ensure that the user has paid for the product.

35. A network-based sales system, comprising:

at least one buyer computer for operation by a user desiring to buy products;

at least one shopping cart computer; and

a shopping cart database connected to the shopping cart computer;

the buyer computer and the shopping cart computer being interconnected by a public packet switched computer network;

the buyer computer being programmed to receive a plurality of requests from a user to add a plurality of respective products to a shopping cart in the shopping cart database, and, in response to the requests to add the products, to send a plurality of respective shopping cart messages over the network to the shopping cart computer each of which comprises a product identifier identifying one of the plurality of products;

the shopping cart computer being programmed to receive the plurality of shopping cart messages, and to modify the shopping cart in the shopping cart database to reflect the plurality of requests to add the plurality of products to the shopping cart; and

the buyer computer being programmed to receive a request from the user to purchase the plurality of products added to the shopping cart and to cause a payment message to be activated to initiate a payment transaction for the plurality of products added to the shopping cart;

the shopping cart being a stored representation of a collection of products, the shopping cart database being a database of stored representations of collections of products, and the shopping cart computer being a computer that modifies the stored representations of collections of products in the database.

36. A method of operating a shopping cart computer in a public packet switched computer network comprising at least one buyer computer for operation by a user desiring to buy products, at least one shopping cart computer, and a shopping cart database connected to the shopping cart computer, the method comprising the steps of:

receiving, at the shopping cart computer, a plurality of shopping cart messages sent over the network to the shopping cart computer by the buyer computer in response to receipt of a plurality of requests from a user to add a plurality of respective products to a shopping cart in the shopping cart database, each of the shopping cart messages comprising a product identifier identifying one of the plurality of products; and

modifying the shopping cart in the shopping cart database to reflect the plurality of requests to add the plurality of products to the shopping cart;

the buyer computer being programmed to receive a request from the user to purchase the plurality of products added to the shopping cart and to cause a payment message to be activated to initiate a payment transaction for the plurality of products added to the shopping cart;

the shopping cart being a stored representation of a collection of products, the shopping cart database being a database of stored representations of collections of products, and the shopping cart computer being a computer that modifies the stored representations of collections of products in the database.

37. A network-based sales system, comprising:

a merchant database comprising a plurality of digital advertisements and a plurality of respective product fulfillment items;

at least one creation computer for creating the merchant database; and

at least one merchant computer for causing the digital advertisements to be transmitted to a user and for causing advertised products to be transmitted to the user;

the creation computer and the merchant computer being interconnected by a public packet switched computer network;

the creation computer being programmed to create the merchant database, and to transmit the digital advertisements and the product fulfillment items over the network to the merchant computer;

the merchant computer being programmed to receive the digital advertisements and product fulfillment items over the network, to receive over the network a request for a digital advertisement from a user, to cause the digital advertisement to be sent to the user over the network, to receive over the network from the user a product request message identifying an advertised product, and to cause the product to be sent to the user in accordance with a product fulfillment item corresponding to the product;

at least a portion of the digital advertisements transmitted by the creation computer to the merchant computer over the network being authenticated by at least one digital signature.

38. A method of operating a merchant computer in a network-based sales system comprising a merchant database that comprises a plurality of digital advertisements and a plurality of respective product fulfillment items, at least one creation computer for creating the merchant database, and at least one merchant computer for causing the digital advertisements to be transmitted to a user and for causing advertised products to be transmitted to the user, the creation computer and the merchant computer being interconnected by a public packet switched computer network, the method comprising the steps of:

receiving, at the merchant computer, the digital advertisements and the product fulfillment items, the digital advertisements and the product fulfillment items having been transmitted over the network to the merchant computer by the creation computer, the merchant database comprising the digital advertisements and the product fulfillment items having been created by the creation computer;

receiving over the network a request for a digital advertisement from a user;

causing the digital advertisement to be sent to the user over the network;

receiving over the network from the user a product request message identifying an advertised product; and

causing the product to be sent to the user in accordance with a product fulfillment item corresponding to the product;

at least a portion of the digital advertisements transmitted by the creation computer to the merchant computer over the network being authenticated by at least one digital signature.

* * * * *